Discovering topics in Slack message streams

Iiro Aalto

Pro gradu
UNIVERSITY OF HELSINKI
Department of Computer Science

Helsinki, June 4, 2020

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author
Iiro Aalto

Työn nimi — Arbetets titel — Title

Discovering topics in Slack message streams

Oppiaine — Läroämne — Subject
Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Pro gradu	June 4, 2020	49

Tiivistelmä — Referat — Abstract

Slack is an instant messaging platform intended for the internal communications of companies and other organizations. For organizations that use Slack extensively it may provide an interesting source of insight, but as such the data is difficult to analyze. Topic modeling, primarily latent Dirichlet allocation (LDA), is commonly used to summarize textual data in a meaningful way.

Instant messages tend to be very short, which causes problems for conventional topic modeling methods such as LDA. The data sparsity problem can be tackled with data expansion and data combination techniques. For instant messages, data combination is particularly attractive as the messages are not independent of each other, but form implicit, and sometimes expicit, threads as the participants reply to each other. Most of the threads in the Slack data are not explicit, but must be 'untangled' from the message stream if they are to be used as a basis for a data combination scheme.

In this thesis we study the possibility of detecting implicit threads from a slack message stream and leveraging the threads as a data combination scheme in topic modeling. The threads are detected using a hierarchical clustering algorithm which uses word mover's distance, latent semantic analysis, and metadata to compute the distances between messages. The clusters are then concatenated and used as the input for LDA. It is shown that on a dataset gathered from the Gofore Oyj Slack workspace, the cluster-based model improves on the message-based model, but falls short of being practical.

ACM Computing Classification System (CCS):

Computing methodologies → Artificial intelligence → Natural language processing

Computing methodologies → Machine learning → Learning paradigms → Unsupervised learning → Topic modeling

Computing methodologies → Machine learning → Learning paradigms → Unsupervised learning → Cluster analysis

Avainsanat — Nyckelord — Keywords
Slack, topic modeling, text clustering

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1 Introduction

Slack is an instant messaging platform designed particularly for the internal communications of businesses and other organizations. At Gofore Oyj, an IT consultancy company of about 500 people, Slack is in heavy use with around 12,000 messages sent daily. The Slack workspace is organized in channels, some of which are public, meaning that they can be accessed by any member of the workspace, and some of which are invitation-only. Discussion on the public channels of the Gofore workspace range from daily work-related issues to various professional and free time interests. The volume and diversity of the discussions makes Slack a good source of data on the interest, concerns and opinions of Gofore employees.

This thesis focuses on summarizing Slack data by identifying topics in the data. Topic models, such as latent Dirichlet allocation (LDA) [7], are often used for such purposes. In topic modeling, the term *topic* is defined as a latent variable which is used to explain differences of word distribution in a set of text documents. While the formal definition is not claimed to match the intuitive concept, the topics tend to be easily interpretable and semantically meaningful. Since LDA and other topic modeling methods rely on word co-occurrences to fit the topics, they run into trouble when the documents in the dataset are short. Since the instant messages in the Slack data tend to be very short, out-of-the-box LDA solution can't be expected to perform well.

In the broad class of text documents, chat messages have several distinct characteristics, the most apparent being their shortness. A single chat message typically contains the worth of a single sentence to a short paragraph of text. This translates to data sparsity which can cause difficulties particularly for bag-of-words methods. However, chat messages are not meant to be read separately but in the context of a discussion. The messages are not independent of of each other, but associated by the explicit structures the messaging platform uses to organize them, and also in the implicit way the participants understand them as part of a discussion.

Syntactically, chat messages are often written in colloquial style, frequently contain misspellings and lack a strict form, with more attention given to expressiveness than correctness. However the style may vary according to the writer's preference; some may choose a more formal style while others prefer to write in a style resembling casual speech. Emojis and emoticons are particularly common, as well as other informal devices used for emotional expression. These syntactical problems add noise and grow the vocabulary, worsening the sparsity issue.

The sparsity issues caused by the shortness of the messages run deep. Term overlap between documents, as well as co-occurrences of terms in the documents carry much information, but they occur less frequently in sparse data; the effects of sparsity for the simple bag-of-words scheme manifest

dramatically in the clustering task presented in Section 6. Large volumes of training data can alleviate the problem, as can heuristically expanding or combining the data. In the expansion strategy, external data is leveraged to add more information in to the dataset. For instance, data from the DBPedia ontology has been used to improve classification [14] and web search results have been used in measuring semantic similarity [30]. When the focus is not on individual messages, it may not be appropriate to consider every message a single data point. Instead, messages may be grouped and combined using metadata or other knowledge on the data. For instance, in analyzing Twitter users, all messages by a single user were concatenated into one document for topic modeling [36].

The data combination strategy is particularly interesting due to the fact that the interpretation of instant messages is so heavily dependent on the context. The interface of a messaging platform informs the participants of the context of a discussion, but does not necessarily reveal it explicitly. If the messages are organized into suitable collections, we can simply change our definition of a document. For instance, in a bulletin board type discussion forum we might consider threads instead of individual messages, since we know that the messages on a thread are strongly connected to one another. In Slack, messages are organized into channels and threads. Depending on user behavior, a discussion might takes place in a single thread, in several threads, on a channel intertwined with other discussions, or even partly in threads and partly on the channel. Nevertheless, the discussions tend to follow a similar pattern: an initiating 'root' message followed by a chain of responses.

It is reasonable to expect that by identifying such discussion, and aggregating individual messages to longer documents accordingly, one might alleviate the data sparsity problem caused by document shortness. This thesis explores solutions for detecting discussions in Slack messages and then leveraging such solutions for deriving better topic models. The main problem of discovering meaningful and informative topics in the data is divided into the sub-problems of discussion detection and topic modeling.

There is little to no research specifically on the Slack platform in the NLP context, but there is some research on IRC and similar platforms [33][1], to which Slack is related. In Slack, messages are organized into channels, some of which are public to all members of the workspace, and some are private. Users can also send direct messages to one another, and reply directly to a message, forming a thread. The data available through the Slack API contains, among other things, the channel ID, thread ID, and the timestamp of each message.

The solution for discussion detection is based on hierarchical clustering. The distance measure for the clustering algorithm is a combination of the word mover's distance (WMD) and latent semantic analysis (LSA). The use of pre-trained word embeddings to calculate WMD also acts as a form of

data expansion, in that it introduces new semantic information to the data from the larger dataset that was used to train the embeddings. Metadata (timestamps and channel identifiers) is used to adjust term weights for improved performance. The presence of threads in the Slack data enables the discussion detection model to be easily tuned and evaluated, assuming that a single thread represents a single discussion.

For topic discovery, the clusters are concatenated into single documents and then used to train an LDA model. The results are evaluated against a baseline LDA model trained on the individual messages. The focus of evaluation is on the topic-word distributions, and their ability to express knowledge about the dataset.

The thesis is structured as follows: Sections 3, 4, and 5 provide relevant background information on text clustering and topic modeling at a general level; Section 6 is dedicated to the sub-problem of discussion detection; in Section 7, the discussion detection is applied to topic modeling, and the final results in the core problem are presented.

2 Topic modeling

Topic models represent a set of text documents using latent variables known as topics, which are used to explain the distribution of words in the documents. In the context of topic modeling, the term *topic* is used in a formal sense; although the concept is motivated by the intuitive meaning of the word, these two should not be conflated. A topic in topic modeling is simply a latent variable, which is used to explain the observed variables, occurrences of words in a set of documents.

At the core of topic modeling is the notion that term occurrences in a document are not independent of each other. The dependencies between term occurrences suggests that the text data can be explained by a much smaller set of latent variables. When applied to text data, the topics reflect semantically meaningful aspects of the documents. Topic modeling algorithms can also be applied to other kinds of discrete data, and have been found useful outside of NLP [6].

Topic modeling originated in information retrieval to answer the need for better relevance estimations between queries and documents [12] by capturing the semantics, rather that lexicon, of text. The earliest form of topic modeling is the non-probabilistic LSA, which is discussed in detail in Section 4. Modern approaches are probabilistic; most notable of these is latent Dirichlet allocation (LDA) [7]. In LDA, each document is associated with a distribution over topics, and each topic is associated with a distribution over terms. The terms of a document are assumed to have been selected by a generative process, in which these distributions are sampled repeatedly.

Topic modeling has a clear relation with (semantic) text clustering. While

topic models do not assign hard labels to documents, the topic distributions of the documents can be seen as a soft clustering. The topic model also provides a soft clustering of the terms in the vocabulary, in the form of the topic-term distribution.

2.1 Latent Dirichlet allocation

Latent Dirichlet allocation estimates the document-topic distributions and topic-term distributions for a set of documents by assuming each document to have been produced by the following generative process: first, the total number of words in the document as well as a topic distribution is chosen; then each individual word is chosen by first sampling a topic from the topic distribution, and then the word from the word distribution given the topic.

In the generative model, documents $\mathcal{D} = \{d_1, ..., d_m\}$ are associated with distributions over topics $\Theta = \{\theta_1, ..., \theta_m\}$, and the topics are are associated with distributions $\Phi = \{\phi_1, ..., \phi_k\}$ over the terms in the vocabulary, where k is the number of topics. Each token d_{ij} in a document d_i is selected by first sampling a topic z_{ij} from θ_i, and then the word from $\phi_{z_{ij}}$. The underlying assumption is that each individual token in the document is explained fully by a single topic, even though the whole document is not. More exactly, a document d_i is assumed to have been generated by the following process:

- Choose N, the number of tokens in d_i
- Choose $\theta_i \sim \text{Dir}(\alpha)$
- For each $j = 1, ..., N$, choose a token d_{ij}:
    1. Choose a topic $z_{ij} \sim \text{Multinomial}(\theta_i)$
    2. Choose a word $d_{ij} \sim \text{Multinomial}(\phi_{z_{ij}})$

LDA is distinct from the earlier pLSA model in that the document-topic distributions Θ are assumed to be Dirichlet distributed. The Dirichlet distribution is controlled by the parameter α, a k-dimensional vector. The topic-word distributions Φ are also assumed to have been drawn from a Dirichlet distribution, controlled by the parameter η.

The assumption that Φ are Dirichlet is a form of smoothing, which is necessary due to the fact that text data is typically sparse; that is, text documents typically do not contain most terms in the vocabulary. This would lead to zero probabilities in an unsmoothed model, which would in turn produce zero probabilities for some unseen documents. In a smoothed model, even unseen cases have a very small but non-zero probability, therefore accounting for rare cases that are not present in the original dataset.

To find the optimal parameters for a given dataset, we would like to maximize the log-likelihood of the data, given by $\sum_{i=1}^{m} \log P(d_i|\alpha, \Phi)$. However,

the computation of $P(d_i|\alpha, \Phi)$ is intractable so it has to be approximated. Approximation methods used for LDA parameter estimation include techniques based on variational Bayes inference, Gibbs sampling, and maximum a posteriori estimation [4].

3 Text clustering

Text, like any other form of data, can be clustered using general-purpose clustering algorithms, given that it can be represented in a suitable format. Finding such a representation for text is far from trivial, however. This section provides an overview of some of the clustering algorithms typically used in clustering text.

Since the family of clustering algorithms is vast, we limit our focus on distance-based clustering. Many general-purpose clustering algorithms are based solely on the distances between data points. Such algorithms include, for instance, K-medoids, DBSCAN, and hierarchical clustering. Such algorithms can be applied to text data granted that a suitable distance measure can be provided. In text clustering, the choice of distance measure is far from trivial and crucially important. Usually the text documents are represented as vectors, and the distances between the vectors are then calculated using a typical distance measure, such a cosine or Euclidean distance. Methods for calculating document distances are discussed in Section 4.

Text clustering and topic modeling are closely related, in that one might see topic modeling as a dedicated form of soft clustering for text data. Topic modeling does not assign hard labels to documents, however, and is equally concerned with the terms as with the documents.

3.1 Notation

The following notation is used throughout the thesis. We are typically concerned with a subset $\mathcal{X}$ of the corpus of all documents $\mathcal{D}$. the ith document of $\mathcal{X}$ is denoted x_i and the ith document of $\mathcal{D}$ as d_i. Each document d_i is a sequence of tokens representing a term from the vocabulary $\mathcal{V}$, and the jth token in d_i is denoted by d_{ij}. The vocabulary $\mathcal{V}$ is a set of terms, where the ith term is denoted by t_i. The number of terms in $\mathcal{V}$ is denoted by $n_{\mathcal{V}}$ and the number of documents in $\mathcal{X}$ and $\mathcal{D}$ by $n_{\mathcal{X}}$ and $n_{\mathcal{D}}$ respectively. Finally, we are often concerned with the document-term matrix of a set of documents, which is denoted by $\mathbf{W}_{\mathcal{X}}$. The ith row of $\mathbf{W}_{\mathcal{X}}$ represents x_i and is denoted by w_i, while the jth weight of w_i is denoted by w_{ij}.

3.2 Hierarchical clustering

Hierarchical clustering is a family of clustering algorithms which produce a tree-like structure instead of a fixed labeling. The tree is formed by merging clusters iteratively based on a similarity criterion. At each iteration, the most similar pair of clusters is merged into a single cluster, until only one cluster remains. This can also be done 'in the opposite direction' by iteratively splitting clusters, starting with all data points in one cluster. The final product either way is a tree-like structure, from which different clusterings can be obtained using various criteria, the most obvious being the desired number of clusters.

The method for calculating cluster similarity is essential in hierarchical clustering, as different methods vary in the clusterings they produce as well as their computational complexity:

- In **Complete linkage**, the similarity of a pair of clusters is determined by the maximum distance between any pair of data points in the clusters. Complete linkage ensures that no two points in a single cluster are very far from each other, forming tightly packed clusters. Given a distance measure d, the distance between clusters U and V in complete linkage is $\max_{u \in U, v \in V} d(u, v)$.

- **Single linkage** is the exact opposite of complete linkage, where the similarity of the clusters is determined by the minimum distance between any pair of data points. Formally, the distance between clusters U and V is $\min_{u \in U, v \in V} d(u, v)$. Single linkage produces clusters that are more akin to an area of density. Single linkage can produce clusters where some points are considerably far from each other. This is called *chaining* and can lead to a low quality clustering.

- In **Average linkage**, the distance between clusters U and V is the average distance of all pairs of points in U and V, that is $\frac{1}{|U||V|} \sum_{u \in U, v \in V} d(u, v)$.

Generally hierarchical clustering has a time complexity of $O(n^3)$, which in most cases is prohibitive. For some linkage methods however, including each of the aforementioned linkage methods, the time complexity can be reduced to $O(n^2)$.

3.3 Other clustering algorithms

The most significant shortcoming of hierarchical clustering is its computational complexity. When efficiency is key, centroid-based clustering algorithms provide a less complex, but also a more limited alternative. Given a desired number of clusters k, centroid-based clustering algorithms iteratively form clusters around k centroids. Perhaps the most commonly known algorithm

in this family is the K-means algorithm, in which the centroids are cluster means. In K-means, the k centroids are first selected randomly from the dataset and assigned to a cluster of their own, and the rest of the data points are then assigned to the clusters with the centroid closest to them. New centroids are calculated by taking the cluster means. Then, the data points are re-assigned and new centroids are computed until optimal centroids are found. Another alternative is the K-medoids algorithm, in which the centroids are always taken from the dataset. [2]

The DBSCAN algorithm [13] is designed to find arbitrarily shaped clusters effectively while relying on hyperparameters as little as possible. Notably, DBSCAN does not require the number of desired clusters to be specified, which is a clear advantage over centroid-based algorithms. In hierarchical clustering on the other hand, there are a number of different ways to form flat clusters from the cluster tree, and the quality of the flat clustering is critically affected by the chosen method. DBSCAN is less reliant on such external choices.

A particular feature of DBSCAN is that it discerns between clusters and noise due to its density-based approach. Given the parameters ϵ and $minPts$, DBSCAN first identifies a set of *core points*, which have a minimum of $minPts$ of points within a distance of ϵ from them. Clusters are then formed around the core points based on connectivity, which is determined by *eachabilityr*: a point p is said to be *directly reachable* from a core point q if the distance between p and q is less than ϵ. A point p is said to be *reachable* from a core point q if there is a path of core points $q_1, ..., q_n$ such that p is directly reachable from q_1, every q_i is directly reachable from q_{i+1}, and $q_n = q$. Finally, a point p is *connected* to a point q if there is such a point o that both p and q are reachable from o. A cluster C is defined by the following conditions: Firstly, if a point p is reachable from q, and $q \in C$, then $p \in C$; and secondly, if $p, q \in C$, then p is connected to q. Due to this definition of a cluster, DBSCAN does not necessarily assign every point in the dataset to a cluster. The remaining points are instead classified as noise.

DBSCAN assumes that all clusters have a similar density, which is determined by the distance ϵ. This is a significant weakness, since the algorithm will not perform well when there are clusters of varying densities. A hierarchical extension of the algorithm, HDBSCAN, has been proposed to solve the problem. [9]

4 Document distances

As such, text is an unstructured form of data, which encodes the information contained in it in a deeply complex way. Transforming the data into an appropriate, structured form is a critical step in finding a working solution for a NLP problem. General purpose clustering methods rely on a distance

measure, and deriving a measure for *document distance*, the distance between text documents, is therefore an essential step in clustering text.

First, we must define what we mean by *document distance*, since the term has no clear and obvious interpretation. For instance, the sentences 'I like cheese' and 'I don't like cheese' talk about the same thing, and on the other hand express a very different sentiment towards it. In this case, we are interested on the topic of a document, on what the document is about, not what it says about it. The distance measures presented in this sections are concerned with general semantic meaning of the documents in relation to other documents.

Document distances can be calculated by first embedding the documents in a vector space and then computing the distance using any conventional measure of vector dissimilarity, such as cosine distance, or by a specialized distance measure such as the word mover's distance. All the methods presented here are based on the bag-of-words assumption in that they discard word order, focusing only on the frequency of words in a document. Note that the terms *bag-of-words* and *vector space model* are often used interchangeably. In this thesis, the term *vector space model* refers to the formally defined model presented in this section, and the term *bag-of-words* to the general principle.

The vector space model (VSM) is a rudimentary method of representing text documents in a vector space. In VSM, a set of documents is represented as a document-term matrix which describes the weight of each term in each document. Relying on exact term overlap, VSM struggles with short documents. Latent semantic analysis (LSA) performs dimensionality reduction on the document-term matrix in order to derive a vector representation that captures the meaning, rather than just the lexicon, of the document. Both VSM and LSA are tried and tested methods that have been used for decades.

A more modern approach relies on Word2Vec word embeddings, which are vector representations for individual words learned from massive data using a neural network model. Representations for whole documents can be derived from word embeddings by aggregation. Embeddings can also be used to compute the word mover's distance, a semantic document distance measure. Word2Vec embeddings can capture complex semantic relationships between words, but their benefits are difficult to translate to entire documents.

4.1 Vector space model

While understanding the precise meaning of a text requires that all complexities of natural language are taken into account, the words alone carry much information. The vector space model [32] represents a text document as a vector of term weights, relying only on term weights to carry the information relevant to the task at hand. As such, the vector space model gives us a rudimentary, yet often well-performing model which is easily usable with

typical, out-of-the-box machine learning methods; on the other hand it has clear limitations as it is oblivious to word order as well as the semantic relationships between the words. The latter becomes more prominent with short documents, which have a smaller chance of term overlap. VSM can still act as starting point for methods that overcome this obstacle.

VSM describes a set of documents as a document-terms matrix, where each column represents a term in the vocabulary. The rows of the matrix represent the documents through the weights associated with each term. Given a set of N documents $\mathcal{D} = \{d_1, ..., d_N\}$ and a vocabulary of M terms $\mathcal{V} = \{t_1, ..., t_M\}$, each document d_i is embedded in a M-dimensional vector space, where the jth component of the vector represents the weight for the term t_j. In this thesis, the vector associated with document d_i is called the *term weight vector*, or simply the term weights, of d_i. Together the term weight vectors for $\mathcal{D}$ form a $N \times M$ document-term matrix $\mathbf{W}_{\mathcal{D}}$, where the rows represents a document and columns the terms, the ith row being the term weight vector for the corresponding document d_i.

The weights of the document-term matrix should reflect the significance of the term in light of the task at hand. The term weighting scheme is of crucial importance for the quality of the resulting representation. Simple weighting schemes include term counts, frequencies, and boolean indicators. All of these schemes assume that all terms are equally significant, which does not tend to be the case. The TFIDF weighting scheme emphasizes terms that appear frequently in the particular document, but rarely across all documents [3]. It does so by normalizing the term frequencies (TF), with their corresponding *inverse document frequencies* (IDF). IDF is argued to reflect the specificity of a term, or how precise its meaning is [34].

The TFIDF weights for a document d_i in $\mathcal{D}$ is given by

$$w_{ij} = TF(t_j, d_i) \cdot IDF(t_j, \mathcal{D}), \tag{1}$$

where $TF(t_j, d_i)$ is the so-called term frequency of t_j in d_i, and $IDF(t_j, \mathcal{D})$ is the inverse document frequency

$$IDF(t_j, \mathcal{D}) = \frac{N}{|\{d \in \mathcal{D} : t_j \in d\}|}, \tag{2}$$

where $|\{d \in D : t_j \in d\}|$ is the number of documents in which the term t_j appears. The 'term frequency' does not have to be an actual frequency, but several different schemes are used. Counts and frequencies are typical choices, but have been criticized in that they assume that a term's importance is proportional to the number of times it appears in the document. Sublinear term frequencies or boolean indicators can be used instead.

Distances between term weight vectors can be computed using measures such as Euclidean distance, or more typically, cosine similarity [3]. The

cosine similarity between vectors v and u is given by

$$similarity(v, u) = \frac{v \cdot u}{\|v\|_2 \|u\|_2}. \quad (3)$$

Cosine similarity can converted into distance by $1 - similarity(u, v)$. Cosine distance is not affected by the magnitude of the vectors, which is usually beneficial since the magnitude of a term weight vector tends to depend mostly on the length of the document.

Since VSM relies on exact term matches, documents that do not share any terms will have maximal distances. This becomes a problem with short documents where the probability of such cases becomes larger. A dramatic manifestation of this problem in clustering is presented in Section 6. Even where there is some overlap, the presence of synonyms and other closely related terms is not reflected in the document distance. To overcome this, the semantic relations between terms need to be integrated in to the representation.

4.2 Latent Semantic Analysis

The overcome the shortcomings of VSM, the semantic relationships between terms need to be considered, not just the terms themselves. While documents that use the same words tend to have the same meaning, it is not necessarily true the other way around. It is often possible to paraphrase a sentence so that the meaning is intact but there are no shared words, stop words excluded. In such cases, the distance between the term weight vectors would not be reflective of the similarity of meaning between the two sentences.

Latent semantic analysis (LSA, also latent semantic indexing, LSI) [12] finds semantic structure in text documents using matrix decomposition. It relies on the assumption that semantically related words often tend to appear in the same context, known as the distributional hypothesis [31]. LSA applies singular value decomposition on a document-term matrix, producing lower-dimensional representations for the documents. The components of the resulting representation represent latent concepts, rather than individual terms or even sets of them. As discussed in Section 2, LSA is the predecessor of topic models such as LDA. However, it is notable that LSA was first conceived in information retrieval as a method of comparing documents, i.e. calculating document distance.

Singular value decomposition (SVD) is a matrix decomposition method, which is typically used for dimensionality reduction in the context of machine learning. Dimensionality reduction tends to lead to some degree of loss of distinction between data points. While the resulting lower-dimensional representation might generally be considered a worse representation because of it, it is exactly what gives LSA its semantic nature. After dimensionality reduction, superficially dissimilar term weight vectors can be very similar or even indistinguishable.

4.2.1 Singular value decomposition

The singular value decomposition of a $n \times m$ real matrix $\mathbf{M}$ is given by

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top, \tag{4}$$

where $\mathbf{U}$ and $\mathbf{V}$ are matrices with orthonormal columns of $n \times r$ and $m \times r$ dimensions respectively, and $\mathbf{\Sigma}$ is a $r \times r$ diagonal matrix, r being the rank of $\mathbf{M}$. In LSA, $\mathbf{M}$ is document-term matrix of n documents and m terms. SVD can be applied on any real or complex matrix, but only the real case is considered here.

The values on the diagonal of $\mathbf{\Sigma}$ are known as the *singular values* of $\mathbf{M}$, and are given by the square roots of the eigenvalues of the matrix $\mathbf{M}^\top\mathbf{M}$, or equivalently of $\mathbf{M}\mathbf{M}^\top$. The columns of $\mathbf{U}$ and $\mathbf{V}$ are given by the eigenvectors of $\mathbf{M}\mathbf{M}^\top$ and $\mathbf{M}^\top\mathbf{M}$ respectively.

While the r singular values of $\mathbf{M}$ are required to reconstruct $\mathbf{M}$ exactly, it is possible to approximate $\mathbf{M}$ by setting all but the k largest singular values to zero, or by removing the corresponding rows from $\mathbf{\Sigma}$ as well as the corresponding columns from $\mathbf{U}$ and $\mathbf{V}$. Doing so yields the matrices $\mathbf{\Sigma}_k$, $\mathbf{U}_k$, and $\mathbf{V}_k$ which give the rank k approximation of $\mathbf{M}$

$$\mathbf{M}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top. \tag{5}$$

Note that $\mathbf{M}_k$ is still a $n \times m$ matrix, albeit of rank k. The k-dimensional representations for the rows of $\mathbf{M}$ (documents, if $\mathbf{M}$ is a document-term matrix) are given by the matrix $\mathbf{U}_k\mathbf{\Sigma}_k$. Similarly, the columns (terms) are represented in k dimensions by the matrix $\mathbf{V}_k\mathbf{\Sigma}_k$.

4.3 Word embeddings

Word embeddings represent words in a continuous vector space, so that the distances between the word vectors correspond to their semantic relatedness. The position of a word embedding in the vector space should capture the meaning of the word in relation to other words. The Skip-gram and Continuous Bag-of-Words (CBOW) algorithms, known collectively as Word2Vec algorithms produce word embeddings based on the context of a word [20]. The algorithms are based on the same assumption as LSA, that the meaning of the word is conveyed by the other words it tends to appear with. Word2Vec embeddings can capture complex semantical relationships between words, as evidenced by the famous example: the vector for *king*, minus the vector for *man*, plus the vector for *woman* falls closest to the vector for *queen*. [1]

Word2Vec algorithms employ a shallow neural network to solve an auxiliary prediction task; predicting the context given the word in Skip-gram,

[1] This example works in practice with both the English and Finnish pre-trained FastText vectors found at `https://fasttext.cc/`.

and vice versa with CBOW. The neural network consists of the input layer, a single hidden linear layer, and a Softmax output layer, which gives the predicted solution to the auxiliary task. After training, the final embeddings are given by the hidden layer; the output layer is not relevant and can be discarded.

Training a Word2Vec model requires plenty of training data. However, the algorithms are purely unsupervised, which means that publicly available, large corpora such as Wikipedia can be used. Also, precomputed embeddings are publicly available, so one might get away without having to train a model altogether. For the FastText implementation for instance, precomputed CBOW embeddings trained on Wikipedia are available for 157 languages [16] as of 2019.

While the original Word2Vec algorithms operate at the word level, an extension [8] which uses sub-word information has been proposed. The extension has been implemented in the FastText library. In addition to entire words, embeddings are computed for character n-grams as well. The motivation behind this is to improve performance particularly for morphologically rich languages, such as Finnish, where the vast number of unique terms would otherwise cause problems.

4.3.1 Definition

The CBOW model estimates word embeddings by predicting a word given the c words closest to it, referred to as its context. Given a vocabulary $\mathcal{V}$ of M words, each word in the context is inputted to the neural network as M-dimensional one-hot vectors, that is, each dimension represents a word in the vocabulary and each vector has exactly one nonzero component. In total, the input then consists of c vectors in M dimensions. The inputs are projected onto a p-dimensional projection layer. A shared weight matrix is used for projecting each of the c vectors in the input. The result is a single, continuous p-dimensional vector, which is similar to a Bag-of-words representation in the sense that it is not affected by word order, giving the model its name.

The Skip-gram model turns the CBOW approach around, predicting the context of a word. The input consists of a one-hot vector representing the focus word. The input is projected onto a projection layer, similarly as with CBOW, only that projection layer now represents only a single word.

Given a sequence of T words $\mathrm{w}_1, ..., \mathrm{w}_T$, CBOW maximizes the objective function

$$\sum_{i=1}^{T} \log p(\mathrm{w}_i \mid C_i), \tag{6}$$

where C_i is the context of w_i, the c nearest words before and after the focus

word, a total of $2c$ words. The objective of Skip-gram is given by

$$\sum_{i=1}^{T} \sum_{-c \leq j \leq c, c \neq 0} \log p(\mathrm{w}_{i+j} \mid \mathrm{w}_i). \tag{7}$$

In the original formulation, the probabilities $p(\mathrm{w}_i \mid C_i)$ and $p(\mathrm{w}_{i+j} \mid \mathrm{w}_i)$ are estimated by a Softmax layer, which is too computationally expensive to be practical [22]. As each word in the vocabulary is an output class of its own, the number of nodes in the output layer is as large as the vocabulary. The number of words in the vocabulary is typically in the tens of thousands, and so the number of weights to be estimated for the output layer is very large. Training the full softmax classifier requires all weights to be re-estimated after each pass, which would make training prohibitively expensive. This creates a need for computationally less expensive approximations.

Using Negative Sampling (NES)[22][21], a purpose-built approximation method, only the weights associated with a handful of words need to be estimated during a single pass. Negative Sampling replaces the softmax layer with a logistic classifier which is trained to separate a real estimated vector from a small number of so-called negative samples. Negative Sampling reduces the multi-class classification task into a binary classification task by training a logistic classifier to distinguish the real focus word from noise. At each pass a number of random words, negative samples, are chosen. These are then fed to the classifier along with the correct focus word.

In the CBOW objective function, the probability $p(\mathrm{w}_i \mid C_i)$ is replaced by

$$\log(1 + e^{-s(\mathrm{w}_i, C_i)}) + \sum_{\mathrm{w}_n \in N_C} \log(1 + e^{s(\mathrm{w}_n, C_i)}), \tag{8}$$

where N_C is a randomly selected set of negative samples. The scoring function s is defined by

$$s(\mathrm{w}, C) = \frac{1}{2c} \sum_{\mathrm{w}' \in C} u_{\mathrm{w}'}^{\intercal} v_{\mathrm{w}}, \tag{9}$$

where the vectors u_{w}, are word vectors for the context words, and v_{w} is the word vector for the focus word. Vectors u_{w} and v_{w} are not the same; these are different parameterizations, representing the projection weights (v_{w}), and the output weights (u_{w}). Similarly, the probability $p(\mathrm{w}_{i+j} \mid \mathrm{w}_i)$ in the Skip-gram objective function is replaced by

$$\log(1 + e^{-u_{\mathrm{w}_{i+j}}^{\intercal} v_{\mathrm{w}_i}}) + \sum_{\mathrm{w}_n \in N_C} \log(1 + e^{u_{\mathrm{w}_n}^{\intercal} v_{\mathrm{w}_i}}). \tag{10}$$

Here, the relationship of u_{w} and v_{w} to the weights of the neural network is more clearly seen.

Appropriate values need to be selected for the hyperparameters; the context size c, the dimensionality of the projection layer p, and the number

of negative samples in N_C. The resulting word embeddings will naturally be p-dimensional as well. Typically, the dimensionality of the embeddings is in the hundreds, and a context size of five is used. Increasing the dimensionality and the context size can improve the quality of the resulting vectors, but will also make training more expensive. For the number of negative samples, 5–20 samples for small datasets and 2-5 for large datasets are suggested.

FastText expands on the original Word2Vec algorithms by estimating vectors for character n-grams, and representing whole words as sums of these. Word-level methods typically struggle with morphologically rich languages, which have a large number of inflected forms for each stem. With the inclusion of subword information in the form of character n-grams, FastText attempts to alleviate the problems caused by morphological complexity. As an additional benefit, FastText can compute word vectors for words not encountered in the training data, as long as the word contains at least one character n-grams that is.

4.3.2 Document distances

Word embeddings represent individual words in a semantically meaningful way, but deriving representations for whole documents from them is a non-trivial issue. Word embeddings are often used as inputs for deep neural networks, but much simpler methods can also yield decent results. The methods discussed here share the Bag-of-words assumption in that they rely on term weights, discarding word order. However, unlike VSM, these methods don't rely on exact term matches, but can capture deeper semantic relationships between documents.

Aggregation Several simple methods for deriving document representations from word embeddings by aggregation have been explored. One such representation is the weighted mean of the embeddings of the terms present in the document. Terms may be weighted with a general-purpose term-weighting scheme such as those presented in Section 4.1. A method for learning optimal weights specifically for short documents has also been proposed [11], but it requires labeled training data.

Given a set of documents $\mathcal{D} = \{d_1, ..., d_N\}$, a vocabulary $V = \{t_1, ..., t_M\}$, and an $N \times M$ document-term matrix $\mathbf{W}$, the weighted mean of the embeddings of the terms in a document d_i is given by

$$\text{MEAN}(d_i) = \sum_{j=1}^{M} \mathbf{W}_{ij} e(t_j), \tag{11}$$

where $e(t)$ is the embedding for term t. The Euclidean distance between the means $\text{MEAN}(d_i)$ and $\text{MEAN}(d_j)$ is known as the Word Centroid Distance (WCD) [19].

Word mover's distance Similarly as the distances between word embeddings are used to measure semantical similarity between words, the word mover's distance (WMD) [19] is used to measure the similarity of entire documents. In WMD, documents are handled as distributions of points in the vector space defined by the word embeddings. The word mover's distance between two documents is the minimal amount of 'work' required to shift one distribution to the other.

The word mover's distance is a special case of a more general problem known as the earth mover's distance (EMD), from which the solution is derived. An intuitive analogue of the EMD problem is finding the least laborious way of filling a number of holes with the sand from a group of piles, assuming that there is enough sand to fill the holes [28]. The earth mover's distance problem was first formulated as far back as 1785, and has been used, with different names, in various fields of research [15]. In machine learning, EMD was first applied in machine vision [28].

Efficient solvers are available for EMD due to its long history, and they can be directly applied to WMD. The best average time complexity of WMD is still of the order $\mathcal{O}(p^3 \log p)$, where p is the number of unique terms in the documents. Computing WMD is therefore quite expensive for large datasets with a high number of unique words. Solving a relaxed version of the WMD problem has a more manageable time complexity of $\mathcal{O}(p^3)$, and yields an approximation known as the relaxed word mover's distance (RWMD).

The starting point for the computation of WMD are the M-dimensional weight vectors w and w' for documents d and d' and the embeddings for the vocabulary $\mathcal{V} = \{t_1, ..., t_M\}$. The weight vectors are normalized to unit-length, so there is always as much 'sand' as is required to fill the 'holes'. The embeddings are used to compute the distance measure $c(i, j)$, which defines the cost of transporting term t_i to term t_j. In the original paper, $c(i, j)$ is defined as the Euclidean distance between the embeddings of the words. The WMD problem is a linear programming problem that asks for a $M \times M$ flow matrix $\mathbf{T}$ that solves the optimization problem

$$\begin{aligned} &\underset{\mathbf{T}\geq 0}{minimize} && \sum_{i,j=1}^{M} \mathbf{T}_{ij} c(i,j) \\ &subject\ to && \sum_{j=1}^{M} \mathbf{T}_{ij} = w_i \quad \forall i \in \{1, ..., M\} \\ &and && \sum_{i=1}^{M} \mathbf{T}_{ij} = w'_j \quad \forall j \in \{1, ..., M\}. \end{aligned} \tag{12}$$

$\mathbf{T}_{ij}$ is the flow from the ith term of w to the jth term of w', that is, how much of word t_i is transported from document d to word t_j of document d'. The first constraint dictates that the flow from a word in d matches its

weight in the document. Similarly, the second constraint requires that the flow to a word in d' equals its weight.

The WMD problem can be relaxed by dropping one of the constraints. Since we now only have to balance the flows in one direction, the problem becomes a nearest neighbor search. This is due to the fact that all the flow from (or to) a word always goes to the nearest word in the other document. It then follows that the solutions to the two versions of the relaxed problem give the distances

$$\ell_1 = \sum_{i=1}^{M} w_i \min_j c(i,j) \tag{13a}$$

$$\ell_2 = \sum_{j=1}^{M} w'_j \min_i c(i,j) \tag{13b}$$

The distance ℓ_1 and ℓ_2 are both good lower bounds of WMD. A better lower bound can be obtained by taking the maximum of these two approximations; the maximum is the relaxed word mover's distance.

5 Evaluation methods for clustering and topic modeling

Precise evaluation of both text clustering and topic modeling is somewhat difficult since they are based on the intuitive and somewhat ambiguous notion of texts sharing similarity on the semantic level. The chosen evaluation method needs to accurately reflect what is considered desirable in the particular task. Various evaluation measures have been developed for both classes of algorithms, but their suitability to any given task needs to be considered carefully, particularly when the ultimate judge is human intuition.

Both clustering and topic modeling are often used to gain insight from a dataset by finding structures in it. The results must then be easy to interpret intuitively. For example, it is often desirable that the topics of a topic model are semantically coherent and capture relevant aspects of the texts. The requirement of interpretability puts a human element into the mix, which might lead to a disparity between what is actually desirable and what a generic evaluation measure says. It is therefore important to understand the underlying assumptions which define what an evaluation measure considers 'good'.

5.1 Clustering

Clustering can be used for very different end purposes, such as exploratory analysis, as a part of a larger task, or to find natural segments in the data. The choice of evaluation method should be informed by the intended purpose

of the clustering algorithm. The measures used for evaluating clustering are typically divided into (at least) the following categories [24]:

- **Internal criteria**, which use only the data itself, and
- **External criteria**, which compare the clustering to a ground truth clustering that represents the ideal solution. A specialized similarity function, several of which have been developed for clusterings, is used for the purpose.

External evaluation criteria can very accurately reflect the quality of a clustering if labeled data is available. This is often not the case however, since such data is not needed in the clustering itself. Only external evaluation is relevant for the purposes of this thesis, and therefore internal evaluation is not discussed.

External evaluation of a clustering is superficially similar to the evaluation methods used in classification, but requires specialized methods. In a clustering task, neither the number of classes nor their labels are known, which rules out the simple measures used for measuring performance in classification. Instead, the clustering is compared to a ground truth clustering using a specialized similarity measure, which does not rely on labels or the absolute number of classes. Such measures can be divided into two categories: pair counting methods, the forefather of which is the Rand index [25][18]; and information theoretic measures, represented here by the V-measure [27] and adjusted mutual information (AMI) [35].

Both pair counting and information theoretic measures are commonly used, as there is no clear preference between the two. It has been suggested that adjusted Rand index (ARI, the adjusted-to-chance version of Rand index) should be preferred over AMI when the clusters are large and evenly sized, and AMI should be preferred when the clusters are unevenly sized. [26]. Whether the measure used should be adjusted to chance is also an important consideration. V-measure and Rand index do not have a fixed baseline to which the result could be compared, and tend to grow as the number of clusters grows. ARI and AMI are normalized by the expected score, making for a more interpretable measure with less bias towards larger number of clusters.

5.1.1 ARI

The Rand index (RI) is a similarity measure for clusterings on the same dataset, based on observing how pairs of data points are related in the two clusterings being compared. Clusterings are said to agree when both assign a pair of points to the same cluster, or both assign them to different clusters. Rand index is the ratio of pairs on which the clusterings agree, of all possible pairs.

Let $U = \{u_1, ..., u_R\}$ and $V = \{v_1, ..., v_C\}$ be clusterings on a dataset with N points. For convenience, let us establish that U is the ground truth and V is the clustering to be evaluated. This is of no real importance however, since the ordering won't change the Rand index. When we draw a pair of points from the dataset at random, one of the following case will apply

I. The points will be in the same class in both U and V

II. The points will be in different classes in both U and V

III. The points will be in the same class in U but in different classes in V

IV. The points will be in different classes in U but in the same class in V

The cases I. and II. are *agreements* and the cases III. and IV. are *disagreements*. Let A be the number agreements and D the number of disagreements. The Rand index of the clustering is given simply by $\frac{A}{A+D}$. The quantity $A + D$ is naturally the number of all possible pairs in the dataset, which is $\binom{N}{2}$. A can be computed using a $R \times C$ contingency table M, where each cell n_{ij} is the number of points that are in the class u_i and the cluster v_j. The sum of the ith row, which is the total number of datapoints in the corresponding class u_i in U, is denoted $n_{.i}$. Similarly, $n_{j.}$ denotes the number of datapoint in v_j.

	v_1	...	v_j	...	v_C	Σ
u_1	n_{11}	...	n_{1j}	...	n_{1C}	$n_{1.}$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
u_i	n_{i1}		n_{ij}		n_{iC}	$n_{i.}$
$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
u_R	n_{R1}		n_{Rj}		n_{RC}	$n_{R.}$
Σ	$n_{.1}$	...	$n_{.j}$	...	$n_{.C}$	

Table 5.1: The contingency table M used for calculating RI. The value n_{ij} in the ith row and the jth column represents the number of data points that are in the ith class in C, and the jth class in K

The number of agreements of type I. and II. are denoted k_{11} and k_{00} respectively and can be computed from the contingency table M. The number of agreements of type I., where the points are in the same class in both clusterings, is given by

$$k_{11} = \frac{1}{2} \sum_{i=1}^{R} \sum_{j=1}^{C} n_{ij}(n_{ij} - 1). \tag{14}$$

The number of agreements of type II. is given by

$$k_{00} = \frac{1}{2}\Big(N^2 + \sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}^2 - \Big(\sum_{i=1}^{R} n_{i\cdot}^2 + \sum_{j=1}^{C} n_{\cdot j}^2\Big)\Big). \tag{15}$$

Similarly, the number of disagreements of types III. and IV. are denoted k_{10} and k_{01} respectively, and given by

$$k_{10} = \frac{1}{2}\Big(\sum_{i=1}^{R} n_{i\cdot}^2 - \sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}^2\Big), \tag{16}$$

$$k_{01} = \frac{1}{2}\Big(\sum_{j=1}^{C} n_{\cdot j}^2 - \sum_{i=1}^{R}\sum_{j=1}^{C} n_{ij}^2\Big). \tag{17}$$

Intuitively, we can see that while the maximum and minimum number of agreements depends on R, C, and the marginal sums $n_{i\cdot}$ and $n_{\cdot j}$ (the sizes of the classes), the exact number of agreements depends ultimately on the quantity $\sum_i^R \sum_j^C n_{ij}^2$. In a case of perfect agreement, each row and column have exactly one non-zero value, which with an appropriate ordering of the matrix fall on the diagonal. Making the clustering less similar, while keeping the cluster sizes fixed, would distribute the 'mass' from the diagonal, and make $\sum_i^R \sum_j^C n_{ij}^2$ smaller. The raw Rand index neither falls on a fixed range in terms of best and worst possible score, nor establishes a baseline so that 'high' and 'low' scores can easily be identified. A more practical score can be obtained by adjusting the index by chance; that is, by fixing it in terms of the expected score of a random clustering and the best possible score. A score of 1 then means that the clusterings are identical, and a score of 0 means that they are no more similar than any random clustering. A measure can be adjusted to chance with the formula

$$\frac{Score - ExpectedScore}{MaximumScore - ExpectedScore}. \tag{18}$$

The adjusted index is much easier to interpret: An above-zero adjusted score indicates a better-than-random clustering, and the best possible score is fixed at one. Adjusted Rand index can be calculated from M by

$$ARI(U, V) = \frac{2 \cdot (k_{00}k_{11} - k_{01}k_{10})}{(k_{00} + k_{01})(k_{11} + k_{01}) + (k_{00} + k_{10})(k_{11} + k_{10})}. \tag{19}$$

according to [35].

5.1.2 V-measure & AMI

V-measure is an information theoretic measure for evaluating clustering, which relies on the *homogeneity* and *completeness* criteria derived from

Shannon entropy. V-measure is defined as the harmonic mean of these two measures. Homogeneity and completeness measure different desirable qualities of a clustering given a ground truth clustering, and can also be used separately for deeper analysis. Unequal weighting is also possible if either aspect is deemed more important. A clustering satisfies the homogeneity criterion when all data points in a cluster are from a single ground truth class. Completeness is satisfied when all data points from a ground truth class are assigned to a single cluster. There tends to be a trade-off between the two criteria; a clustering solution where all data points would be assigned in a cluster of its own would be fully homogeneous (since all clusters would contain data points from a single class), but sacrifice completeness (since all points from a single class would be in different clusters, assuming there are classes with more than one point). Similarly, a clustering with all data points in a single cluster would fully satisfy completeness at the expense of homogeneity. These two degenerate clusterings are both clearly undesirable and underline the insufficiency of either criterion alone. However when the clustering and the ground truth classes are equivalent, both criteria are met.

Homogeneity and completeness are derived from conditional entropy, which is intuitively the amount of information in a distribution that can not be explained by the information in a given second distribution. The conditional entropy $H(X|Y)$ equals the entropy of X when the distributions are independent, and zero when the value of X always follows exactly from the value of Y. The homogeneity criterion is satisfied when the conditional entropy of the ground truth class labels given the cluster labels is maximal, since the class label is always determined by the cluster label. Similarly, the completeness criterion is satisfied when the conditional entropy of the cluster labels given the class labels is maximal, as the cluster follows from the class label.

The homogeneity and completeness scores for a clustering V given ground truth classes U can be calculated using the contingency table M defined in Section 5.1.1. V-measure, like the Rand index, does not depend on which clustering we define as the ground truth. The same is not true for completeness and homogeneity, as they would 'switch places'. For the correct interpretation of the scores it is therefore necessary to explicitly define which clustering is the ground truth.

The homogeneity score h is given by

$$h = \begin{cases} 1 & \text{if } H(U) = 0 \\ 1 - \dfrac{H(U|V)}{H(U)} & \text{otherwise} \end{cases} \tag{20}$$

where $H(U|V)$ is the conditional entropy of U given V, and the normalization term $H(U)$ is the entropy of U. [2] In the upper case there is only a single

[2]Note that the definition is slightly different from [27], where the is some vagueness;

class, and therefore the clustering is guaranteed to be fully homogeneous. Normalization is needed to bound the score within the fixed range $[0, 1]$, as the absolute value of $H(U|V)$ is dependent on the entropy of U, and therefore on the size of the dataset and the classes. The terms $H(U|V)$ and $H(U)$ are given by

$$H(U|V) = -\sum_{j=1}^{C}\sum_{i=1}^{R}\frac{n_{ij}}{N}\log\frac{n_{ij}}{\sum_{k=1}^{R} n_{kj}} \tag{21a}$$

$$H(U) = -\sum_{i=1}^{R}\frac{\sum_{j=1}^{C} n_{ij}}{N}\log\frac{\sum_{j=1}^{C} n_{ij}}{N} \tag{21b}$$

Similarly, the completeness score c is given by

$$c = \begin{cases} 1 & \text{if } H(V) = 0 \\ 1 - \dfrac{H(V|U)}{H(V)} & \text{otherwise} \end{cases} \tag{22}$$

where

$$H(V|U) = -\sum_{i=1}^{R}\sum_{j=1}^{C}\frac{n_{ij}}{N}\log\frac{n_{ij}}{\sum_{k=1}^{C} n_{ik}} \tag{23a}$$

$$H(V) = -\sum_{j=1}^{C}\frac{\sum_{i=1}^{R} n_{ij}}{N}\log\frac{\sum_{i=1}^{R} n_{ij}}{N} \tag{23b}$$

Finally, the V-measure of the clustering is given by the harmonic mean of the homogeneity and completeness scores:

$$V = \frac{2hc}{h + c}. \tag{24}$$

As mentioned earlier, the mean can be weighted towards either criterion, but typically they are given an equal weight, as in the formulation above.

The most significant drawback of the V-measure is that it is not adjusted to chance, and therefore does not provide a stable baseline independent of the number of clusters. As the number of clusters grows, so does the V-measure. However, the adjusting does not change the rank-order of different clusterings with a fixed number of clusters. When adjustment for chance is required, AMI can be used instead. AMI is the adjusted-for-chance version of normalized mutual information (NMI), which is identical to V-measure when homogeneity and completeness are weighted equally [5]. When comparing clustering with the same number of clusters against a ground truth, AMI and NMI only differ in terms of normalization. In such a case, V-measure and AMI therefore rank the clusterings identically. However, if the clusterings to be compared differ in the number of clusters, AMI should be used.

the definition presented here matches the *sklearn* implementation, which was used for the results presented in the later sections. See `https://scikit-learn.org/stable/modules/generated/sklearn.metrics.homogeneity_completeness_v_measure.html`

5.2 Topic modeling

Topic models are difficult to evaluate much for similar reasons as clusterings. The choice of the evaluation method must be informed by the intended purpose of the topic model. For instance, topic modeling might be used as an intermediary step in a larger task, in which case performance in that task is the ultimate measure of quality. However, it is common that the topic model itself is the end product. Very commonly, topic modeling is used to find structure in a corpus. Quality is then not measured by the validity of the model from a statistical perspective, but by its ability to capture relevant information about the data in a way that is easy to for a human to interpret. Evaluating the relevance and interpretability of a topic model quantitatively is not a straightforward task, and often relies on some form of external evaluation, either in the form of human input or semantic data.

The oldest method of evaluating a topic model from a statistical perspective is to measure its ability to predict unseen data by calculating the log-likelihood or *perplexity*[7] of a held-out test dataset given the topic model. However it has been shown that log-likelihood and perplexity do not always reflect human judgment [10]. This is due to the fact that the validity of a topic model from a purely probabilistic perspective does not reflect the actual intended purpose of the topic model.

Human judgment is the golden standard for evaluating topic models when interpretability and relevance are key. The word and topic intrusion tests have been proposed for quantitative human evaluation, which involve random manipulations to the topic model; the subjects must detect intruding words inserted into topics, and intruding topics inserted into the topic assignments of documents. If the topic model is good, the subject should be able to correctly identify these intrusions by their random nature. However if the topics are counterintuitive or difficult to understand, the subjects will be less likely to identify the intruder.

Human evaluation is often too expensive or laborious to be feasible, but fully automated alternatives exist. Coherence measures evaluate a topic model from a semantical perspective by examining the internal coherence of the topics as sets of words. Coherence measures are concerned with the most relevant terms in a topic-word distribution, discarding the probabilities and examining a topic simply as a set of words. Such a set is said to be coherent, if the words support each other; that is, they increase each others likelihood.

5.2.1 Probabilistic methods

Much as one would evaluate a supervised model on a separate validation dataset, a topic model can be evaluated using a holdout dataset. By calculating the likelihood of the holdout, the topic model's predictive power can be evaluated.

Given a training dataset $\mathcal{D}$, topic-word distributions $\Phi = \{\phi_1, ..., \phi_k\}$, and a holdout $\mathcal{D}' = \{d'_1, ..., d'_m\}$, the likelihood of D' given Φ and α is given by

$$P(\mathcal{D}'|\Phi, \alpha) = \prod_{i=1}^{m} P(d'_i|\Phi.\alpha), \tag{25}$$

As the topic distributions are independent of one another, Φ and α are sufficient to estimate the likelihood of $\mathcal{D}'$ given the LDA model. As mentioned in Section 2.1, exact calculation of the document likelihood $P(d'_i|\Phi, \alpha)$ is not tractable, but approximation is. With approximate document likelihood, different evaluation measures can be calculated. The log-likelihood is given by

$$\text{log-likelihood}(\mathcal{D}') = \sum_{i=1}^{m} \log P(d'_i|\Phi, \alpha). \tag{26}$$

Perplexity is a commonly used measure in NLP, and is given by

$$\text{perplexity}(\mathcal{D}') = \exp\left[- \frac{\text{log-likelihood}(\mathcal{D}')}{\sum_{i=1}^{m} N_i} \right], \tag{27}$$

where N_i is the number of tokens in d'_i [7]. Clearly perplexity follows from log-likelihood, so the measures always agree.

Log-likelihood and perplexity accurately measure the quality of the topic model as a generative probabilistic model from a statistical perspective. Due to the way LDA is typically used however, they do not correlate strongly with human judgment, and in the worst case may even contradict it [10].

5.2.2 Human evaluation

Human judgment of a topic model can be tested quantitatively with the *word intrusion* and *topic intrusion* tests, which measure the interpretability of the topics themselves, and the relevance of the topics in light of the data [10].

In the word intrusion test, the topics are presented to the test subjects as sets of their most probable words, with a single random word inserted into each topic. The test subjects are tasked with detecting the randomly inserted intruder. If the topic is easily interpretable, the test subjects should be able to discern a meaningful pattern behind the words and therefore correctly identify the intruder by its distinctive unrelatedness to the others. On the other hand, if the test subjects find no such pattern, they must choose at random since all of the words seem to them equally unrelated. A higher likelihood of test subjects correctly identifying the intruder is therefore considered to indicate a better model.

In the topic intrusion test, the test subjects are presented a snippet from a document in the dataset, and four topics associated with it. Three of the topics are the most probable ones associated with the document, and the

remaining topic is an intruder chosen randomly from the other topics in the model. The test subject is again tasked with detecting the intruder. Similarly as with word intrusion, if the true topic assignments seem relevant to the snippet the test subject should have no trouble identifying the intruder.

Since human evaluation is always expensive, there is a need for fully automated evaluation methods that correlate strongly with human judgments. The word intrusion test has been automated with a high correlation to human judgments, but in automated topic intrusion the correlation is much lower [17].

5.2.3 Coherence

A more direct, fully automated approach to measuring the interpretability of topic models is to use a coherence measure. The concept of coherence comes from the philosophy of science, where a set of facts is considered coherent if the facts support each other; that is, if they make each other more probable. Words can support each other in a similar fashion, and by interpreting words as facts the general concept of coherence can be applied to sets of words [29]. For instance, the set $\{dog, cat, rabbit\}$ would be considered highly coherent, since the words follow a pattern in that they all refer to animals and therefore explain each other, while the set $\{dog, house, moon\}$ has low coherence since there is no discernible relation between the words.

Coherence measures can be used to evaluate topic models by considering the topics as sets of their most probable words, similarly as in word intrusion. Coherence does not consider the quality of the topic assignments to documents at all, and therefore does not measure the topics' relevance in light of the data. Coherence can therefore only be used to evaluate the interpretability aspect of a topic model, not the quality of the topic model as a whole.

Several different methods for calculating coherence have been proposed, but all of them share a similar structure. The family of coherence measures has been unified in a formal framework [29], which defines a coherence measure through four dimensions. The various coherence measures differ in the components of each dimension, but have the same basic structure. Each of the components are interchangeable, so that new coherence measures can be easily constructed.

The outline of a coherence computation is as follows: A set of words is first segmented into pairs of subsets. Then, a confirmation measure is used to compute the confirmation, or how much a subset supports another, for the pairs. For this, word probabilities need to be estimated. Finally, the confirmations for the pairs are aggregated into a single value.

The first dimension of a coherence measure is the segmentation strategy. All coherence measures consider pairs of subsets at a time, but these pairs can be formed any number of ways. The task of forming these pairs is the responsibility of the segmentation strategy. Given a set of words W, a

segmentation strategy produces a segmentation $S = \{S_1, S_2, ...\}$, where each member is a pair $S_i = (W', W^*)$, where $W', W^* \subseteq W$. Note that the pair is ordered. The most typical strategy is to consider all pairs of individual words, known as the *one-one* segmentation.

The second dimension is the method of probability estimation. Joint probabilities for sets of words can be estimated by counting their co-occurrences in a corpus. There are different ways for counting co-occurrences; a typical way is to simply count the times the words occur in the same document, known as the *boolean document* method. Another popular way is the *boolean sliding window* method; the times the words occur within a fixed distance (in words) of each other are counted.

The third dimension is the confirmation measure. The confirmation measure is used to calculate how much a subset supports another using the estimated word probabilites. In the process of computing coherence, the confirmation measure is used to calculate confirmations for all pairs of subsets in the segmentation S. Notable confirmation measures include the pointwise mutual information (PMI) and the normalized PMI (NPMI), which are given for a pair $S_i = (W', W^*)$ by

$$\mathrm{PMI}(W', W^*) = \log \frac{P(W', W^*) + \epsilon}{P(W')P(W^*)}, \tag{28}$$

$$\mathrm{NPMI}(W', W^*) = \frac{\mathrm{PMI}(W', W^*)}{-\log(P(W', W^*) + \epsilon)}. \tag{29}$$

Confirmation measures that use the word probability estimates of the compared sets alone to calculate the confirmation are called *direct* confirmation measures. The problem with direct confirmation measures is that some words may occur rarely together, while still having related meanings. However, such words should still appear in similar contexts, that is they should appear frequently with the same words. They would therefore have a low direct confirmation with each other, but their confirmations with the other words in the set would be similar. This fact is leveraged by *indirect* confirmation measures, which compute a *context vector* for each of the compared subsets. The context vector of a subset W' of W is formed by first computing a direct confirmation for W' with every word in W. The confirmation are combined into a vector v so that the jth element of v is the confirmation of W' and w_j, the jth term in W. More formally, given a direct confirmation measure m, the context vector for W' is given by

$$v_m(W') = \left\{ \sum_{w_i \in W'} m(w_i, w_j) \right\}_{j=1,...,|W|}. \tag{30}$$

Any direct confirmation measure can be selected as m. The final confirmation for a pair of subsets can be calculated with any conventional measure of

vector similarity, such as cosine similarity. A dedicated segmentation strategy can be used with a indirect confirmation measure, known as the *one-set* strategy. The one-set strategy compares each individual term with the entire set W.

The final dimension is the aggregation method. The confirmation measure produces confirmation values for each pair in S, and these need to be aggregated to produce the final coherence score. The most typical aggregation method is the arithmetic mean.

For example, the UCI coherence is produced when using the one-one segmentation strategy, boolean sliding window probabilities estimated from a large corpus such as the English Wikipedia, and PMI or NPMI confirmations aggregated by arithmetic mean.

According to [29], the coherence measure with the highest correlation with human judgments uses probabilities estimated from Wikipedia using the boolean sliding window method, one-set segmentation and indirect NPMI confirmations, aggregated by arithmetic mean. This coherence measures is denoted C_V, for lack of a better name.

6 Detecting discussions

Instant messages often resemble utterances in a face-to-face discussion, and therefore rely on the context to communicate information reliably. The participants are able to discern the context and understand each other effortlessly, but taken out of context the meaning of an individual message becomes obscure. Like face-to-face discussion, the discussions on an instant messaging platform flow by participants replying and reacting to earlier messages. However, unlike face-to-face discussions they can take place across a much longer time span. Participants do not rely on exact temporal information (other than ordering) to infer the context. Adjacency in the temporal ordering is not required either, as the discussion are often 'tangled' in the message stream with several discussions taking place simultaneously. Finally, the discussion usually has a clear starting point, a message, from which it evolves and branches off.

Instant messaging platforms provide various features to organize messages in explicit structures, which provide some indication of the context. However users use the features the way they see fit, which has to be taken into account when working with such data. In Slack, users can post messages to a channel, or reply directly to a message forming a thread. While most messages are not posted in a thread, it is reasonable to assume that they still follow a similar, implicit structure. The detection of such structures in message streams has been studied previously [33]. These structures have been termed *threads*, but to avoid ambiguity we call them *discussions* in this thesis, while the term *thread* is reserved for the explicit structures present in the Slack data. In

previous work, a discussion is defined by a root message and replies to it.

By requiring that a root message is clearly not part of another discussion, the definition is clear. However, these could lead into meandering discussions that may not be ideal for our purposes. Therefore we prioritize certain aspects of an ideal solution over others. Since we are detecting discussions for the purpose of combining individual messages into less sparse but still coherent units of text, it is tolerable that a discussion is detected in several parts. Mistaking several unrelated discussions as a single discussion is much more problematic since the resulting unit of text is not internally coherent. However if several, highly similar discussions are mistaken as one, the result may still be coherent. In fact, if the actual discussions are very short, this may be beneficial. Due to these considerations, we prefer firstly discussions that are internally coherent, and secondly long enough to effectively alleviate the sparsity issue. The completeness of the discussions, that a true discussion is detected as a whole, is a lesser concern.

The problem of detecting discussions is related to the problems of topic detection and tracking (TDT). The word 'topic' has a very different meaning in TDT than in topic modeling. TDT is concerned with recurring themes in an incoming stream of documents; a good example is the detection of events in a stream of news articles.

In order to disambiguate between the various meanings of the key concepts in this thesis, the following definitions are used:

- *Thread* refers to the explicit structure found in the Slack data. A thread consists of a root messages and messages posted as a reply to it. As the root message is always posted on a channel, a thread is a substructure of a channel.

- *Discussion* refers to an implicit thread. Every message is either a root message or a reply and can therefore be assigned to a discussion. It is possible however, that a discussion consist of a single message. For the purposes of this thesis it is not necessary to establish the structure of the discussion; that is, which is the root message and how the replies relate to each other. Rather, a discussion is simply a group of messages.

- *Topic* has different formal definitions in topic modeling and TDT. In this thesis, the topic modeling definition is used. Topic modeling was discussed in Section 2.

In this thesis, we are ultimately interested in summarizing a stream of Slack messages by extracting topics from it. For this purpose, the discussion as such provide little value. Instead they are used to improve results in a topic modeling. We explore, whether detecting these implicit structures is a useful data combination scheme for topic modeling. The motivation for this approach is practical. Firstly, dedicated topic models for the kind

of data we are dealing with are not readily available; by first solving the auxiliary task we hope to effectively apply a conventional topic model for the task. Secondly, labeled evaluation data for the discussion detection task is available, which makes fully automated external evaluation possible. This in turn makes model selection and tuning easier and more reliable. Evaluating the topic model is more difficult, and therefore it would be useful to shift the burden of repeated evaluation in the tuning phase to the discussion detection task.

6.1 The Slack dataset

The approach presented in this thesis is intended for Slack data, particularly data from the Gofore Oyj Slack workspace. Data is available from the course of about two years, and the target is to analyze a single months data at a time, which might be anywhere between 5 000 to 12 000 messages depending on the month.

The data from the Gofore Oyj workspace shares the qualities typical to instant messaging data: sparsity and noisiness. However the data has its own special characteristics that need to be considered as well. Since the company is international but mostly based in Finland, both English and Finnish are used. The language of each message is detected using the Polyglot library for Python. The language detection is not perfect even if messages with unreliable predictions are left out, leading to noise in the dataset in the form of wrong language labels, as well as messages that contain both languages. The dataset is then quite noisy, but on the other hand the messages are of better quality in terms of correctness and length than, for instance, tweets from Twitter.

In order to use external measures for evaluating performance in the clustering task, a manually annotated ground truth dataset would be ideal. Such data is not available, however. Instead, messages posted in threads are used as the ground truth set, with the thread IDs acting as the class labels. Threads are formed in Slack when users reply to a message directly; messages in a thread therefore form a single discussion by the definition presented earlier, as the are all replies to the same root message. However, it is possible that there are several threads that in fact form a single discussion. It is also common that a thread only includes part of a discussion, since users may choose to reply on the channel instead of the thread.

To better reflect the actual task, the whole dataset, including the messages not posted in threads, is clustered. Evaluation measures are then calculated on the subset with ground truth class labels (thread IDs). Several months of data was available, and separate dataset were used for tuning, testing, and finally for the results presented in this thesis.

The evaluation dataset contains all public messages from the Gofore Oyj workspace from the course of April 2018. The dataset contains 5456

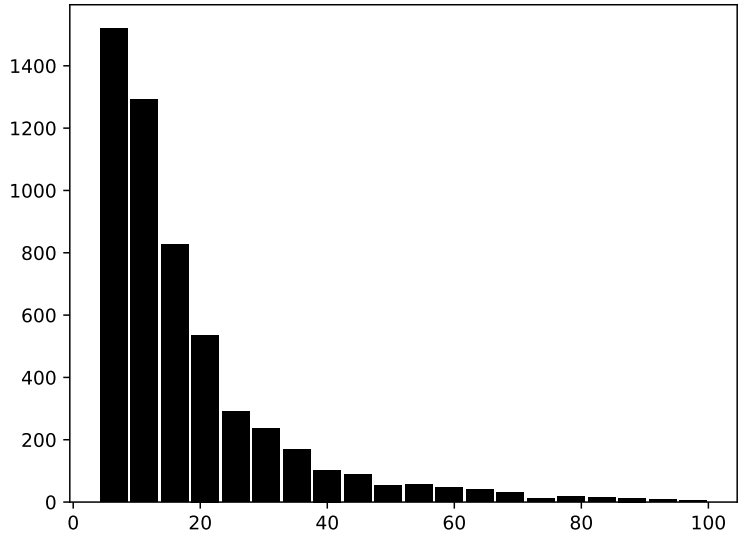

Figure 6.1: The distribution of word counts in the documents in the evaluation dataset

messages, 1321 of which are assigned to a thread; there are 366 threads. The number of messages in a thread varies from a single reply to dozens of messages. The documents are mostly very short, 21 words by average, the median being 13 words. The distribution of word counts can be seen in further detail in Figure 6.1.

6.2 Approach

The solution presented here approaches the discussion detection problem in an unsupervised manner, using complete linkage clustering. Several methods for calculating the distances between documents are considered.

In addition to complete linkage, single and average linkage hierarchical clustering, DBSCAN, and HDBSCAN were considered. Single and average linkage ran into severe chaining issues, resulting in a very poor clustering. DBSCAN and HDBSCAN also performed poorly. Complete linkage clustering was the only clustering method of the ones explored that proved suitable.

6.2.1 Weighting scheme

Two different term weighting schemes are considered. The *pure* weighting scheme uses TFIDF weights, trained on messages from the course of 18 months. Only messages in English are included. The features include

unigrams and bigrams; for the FastText-based models, only unigrams are used, since the precomputed vectors do not contain bigrams.

Filtering out features has a strong effect on the performance. Despite IDF weighting, clusters often form around very common terms, leading to an uninformative clustering. The stop words are determined partly in a manually specified list [3], and partly by filtering out terms by document frequency. Very rare terms tend to be typos or rare bigrams, and are filtered out based on absolute count. A minimum of five occurrences and a maximum document frequency of 0.01 are required. These values were selected based on manual tuning. Note that the maximum document frequency is rather low, as the documents are so short; a typical document contains only a handful of terms, and even the most common terms therefore appear only in a small portion of documents.

The *context* weighting scheme is based on the *pure* weighting, but emphasizes terms that appear with the neighbors of a message. The neighbors of a message share the same channel and are sent within a given time interval from the focus message. The messages in the neighborhood are weighted by their time difference with the focus message and averaged. Terms that do not appear in the focus message are discarded, and the remaining term weights are summed with the focus message term weights.

The effect of the metadata-based weighting on performance is difficult to evaluate using the threads, since the threads do not accurately reflect the entire dataset when it comes to the metadata. The frequency of messages is much lower in the threads subset than in the entire dataset, which has an effect on hyperparameter selection. In terms of the timestamps the threads are not at all comparable to a random sample, since individual threads appear as isolated 'islands' in the data. Emphasizing the metadata over and beyond the actual optimum might therefore appear to increase performance, as these 'islands' are clustered together simply based on metadata. In the entire data, separate discussions are much more likely to appear closely adjacent or intertwined with each other, and a too high emphasis on metadata may have a negative effect on discerning between such discussions. Due to these concerns, a fairly conservative approach of only including the terms that are present in the focus message was chosen.

For a set of documents $d = \{d_1, ...\}$ we have $channel_i$, the channel ID of the message d_i, and ts_i, the UNIX timestamp. Message d_j is in the neighborhood of d_i if and only if $channel_j = channel_i$ and $|ts_j - ts_i| \leq maxdt$, where $maxdt$ is the maximal allowed time difference between d_i and d_j in seconds. The set of indexes of the documents in the neighborhood of d_i is denoted N_i. A context vector c_i for d_i is computed by taking the mean of

[3] The list of stop words is from *sklearn* and can be found at `https://github.com/scikit-learn/scikit-learn/blob/master/sklearn/feature_extraction/stop_words.py`.

the term weights for the documents in the neighborhood, weighted by their time difference with d_i. The weights for the documents in the neighborhood are given by

$$v_i = \big(-(|ts_j - ts_i| - maxdt)\big)_{j \in N_i}. \tag{31}$$

The context vector c_i is then given by

$$c_i = \frac{\sum_{j \in N_i} v_{ij} w_j}{\sum_{j \in N_i} v_{ij}}, \tag{32}$$

where w_j are the the term weights for document d_j.

The modified term weights w_i' for d_i are calculated from the original term weights w_i and the context c_i as follows:

$$w'_{ij} = \begin{cases} 0 & \text{if } w_{ij} = 0 \\ (1-\alpha) w_{ij} + \alpha c_{ij} & \text{otherwise} \end{cases} \tag{33}$$

Finally, w_i' is normalized to unit length using the ℓ^2 norm.

6.2.2 Distance measures

Three different methods for calculating document distance were considered. The methods have their own distinct advantages and disadvantages, and in an effort to find a more balanced measure, combinations of the measures were also considered.

The methods based on word embeddings use the FastText[4] library with 300-dimensional embeddings trained on the contents of the English Wikipedia. The pre-trained embeddings are not normalized by default; in this case, the vectors are always normalized to unit length.

LSA The measure denoted by **LSA** is calculated by first reducing the dimensionality of the term weight vectors to 500 using LSA. The relatively large number of components needed is most likely due to the sparsity. Cosine distance is then used to measure document distance. Note that **LSA** uses bigram features, unlike the other measures.

WMD Word mover's distance is computed using the cosine distances of the embeddings as the distances of individual words. The original article assumes the distances to be Euclidean, but cosine distances outperformed Euclideans in this case. A significant drawback of WMD is its time complexity, which in this case is within the tolerable range, but would not scale to much larger datasets. On a up-to-date laptop computer, computing the full distance matrix for a dataset of less than 10,000 documents takes up to 50 minutes.

[4]Both the FastText library and the pre-trained embeddings are publicly available at `https://github.com/facebookresearch/fastText`

The implementation is not parallelized, but given that the size of the distance matrix grows quadratically, computing it would become intractable with larger datasets. The WMD measure is denoted by **WMD**.

Aggregated FastText Since WMD is computationally so expensive, the aggregated FastText approach as presented in section 4.3.2 was explored as a substitute. For each document, the word centroid distance (WCD) is calculated; that is, the mean of the embeddings for each term in the document weighted by the corresponding term weights. Document distances are then given by the cosine distances of the means. The measure is denoted by **WCD**.

Combined measures The **LSA**, **WCD**, and **WMD** models all produce quite different clusterings, each with their own distinct strengths and weaknesses. Like averaging outputs of several classifiers can increase performance in a classification task, using a linear combination of different distance measures can yield better results in clustering [23].

The combined measure **Blend** of $\delta_a$and δ_b is given by

$$\textbf{Blend}(d_i, d_j) = (1 - \alpha)\delta_a(d_i, d_j) + \alpha\delta_b(d_i, d_j), \tag{34}$$

where $0 \leq \alpha \leq 1$. The blend ratio α defines the weight of each measure and must be manually tuned. For the results shown here, α was chosen by calculating the average ARI over a range of cluster counts. The performance of the combined measure seems to be quite sensitive to α, but wide range of values improves on both measures. Also, the combined measure always seems to perform at least as well as the worse of the two measures. Both **Blend**(**LSA**, **WMD**) and **Blend**(**LSA**, **WCD**) performed better than any single measure. These are denoted by $\textbf{BLEND}_{\textbf{WMD}}$ and $\textbf{BLEND}_{\textbf{WCD}}$ respectively.

6.3 Results

In this section, the performance of the **WMD**, **WCD**, and **LSA** measures and the combined measures $\textbf{BLEND}_{\textbf{WMD}}$ and $\textbf{BLEND}_{\textbf{WCD}}$ are analyzed in the sub-problem of discussion detection. The clustering algorithm is ultimately used as a middle step in a topic modeling task, and the ultimate measure of quality for the clustering algorithm is the performance gain in the primary task. However, in this section we treat the discussion detection as a separate problem, and evaluate it as such.

The clustering algorithm is evaluated using the external measures ARI and V-measure. Though manually labeled evaluation data is not available, thread IDs are present in the Slack dataset and are used as the ground truth dataset instead. Most of the messages in the dataset do not have thread IDs,

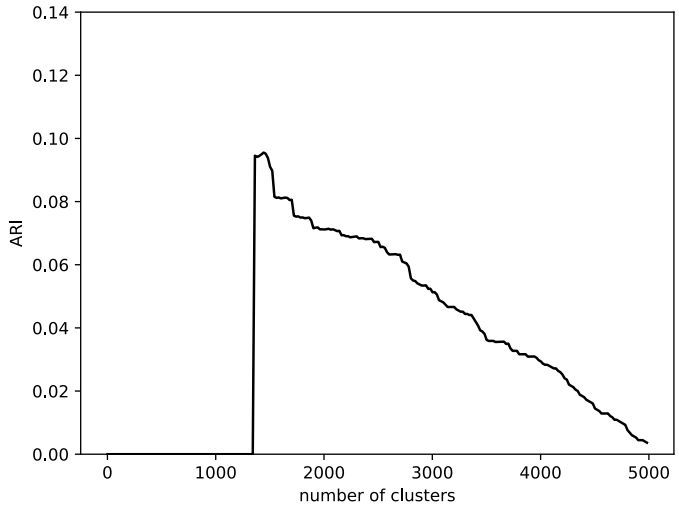

Figure 6.2: The plain vector space model manages to produce clusters only when the number of clusters is sufficiently high.

and therefore the accuracy of their cluster assignments cannot be evaluated using external measures. The threads alone do not represent the dataset accurately in terms of metadata, however, and therefore the whole dataset is clustered. ARI and V-measure are then calculated on the subset of the data with thread IDs.

As previously discussed, the plain vector space model runs into trouble with very short documents. In Figure 6.2, the cosine distances of the term weight vectors are used to compute the cluster hierarchy. The cutoff at about 1500 clusters is due to the fact that at that point, the distances between all pairs of clusters are maximal. That is, if any pair of clusters was merged the resulting clusters would have a pair of documents with the maximal document distance. The complete linkage clustering algorithm cannot therefore produce a flat clustering without merging clusters arbitrarily when the number of desired clusters is below the cutoff point. The problem affects complete linkage clustering in particular, as merging is based on the maximal distance between any pair of documents; all pairs are therefore required to have some term overlap in order to be clustered together. Single and average linkage clustering are not as severely affected, but are unsuitable due to the chaining issues mentioned earlier.

The scores are presented here for a wide range of cluster counts. All cluster counts are not equally important for our purposes, however. If we expect a similar ratio of discussions to messages as there are in the threads, we would expect around 1500 clusters to be suitable. The number may be

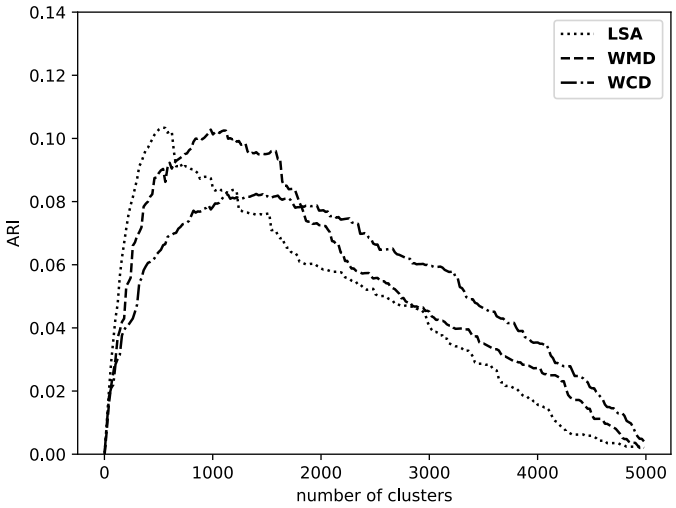

Figure 6.3: The ARI scores of the **WMD**, **WCD**, and **LSA** models.

too high however, as the threads may represent only partial discussions. In Figure 6.3 we see that the performance of each distance measure peaks in the range of 500 to 1500 clusters. Manual inspection of the clusters reveals that when the number of clusters is very low, clusters tend to become internally less coherent, being combinations of several loosely related sub-clusters. On the other hand, when the number of clusters is very large, the clusters become needlessly fractured. A wide range of cluster counts produce decent clusterings, however. Based on manual inspection, the number of clusters should be at least in the hundreds in order for them to be coherent. When the number of clusters is in the thousands on the other hand, the clusters are very small, with many comprising only of a single message. For model selection and tuning, the focus range was set to 200 to 2000 clusters.

The most notable result is that combining measures improves performance significantly. In Figure 6.3, we can see that **LSA** performs well when the number of clusters is low, but **WMD** quickly overtakes it at about 700 clusters. However the combination of **LSA** and **WMD**, $\mathbf{BLEND_{WMD}}$, outperforms both measures across the board as can be seen in Figure 6.4. The **WCD** measure performs clearly worse than **WMD** in the significant range of cluster counts, which translates to the performance of $\mathbf{BLEND_{WCD}}$. The $\mathbf{BLEND_{WCD}}$ still outperforms every individual measure overall, although we can see that **LSA** has a very slight advantage over it when the number of clusters is low. Curiously, the **WCD** measure performs better than any other measure when the number of clusters is over 2000; this is insignificant for our purposes, however.

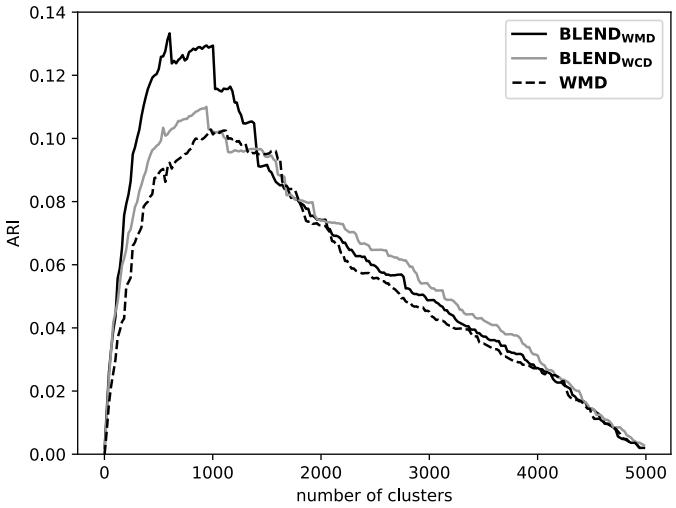

Figure 6.4: The performance of the **BLEND$_{WMD}$** and **BLEND$_{WCD}$** models, compared to **WMD**.

There are significant differences on how the **LSA** measure and the word embedding based measures behave. The **LSA** measure produces topic-like clusters, being at its best when the number of clusters is relatively small. However, the performance drops sharply as the clusters get smaller, since the small clusters are not particularly coherent. The **WMD** measure on the other hand produces very coherent small clusters, but when the number of clusters is small the clusters become very unevenly sized and not particularly meaningful. The large clusters produced by **WMD** do not have a topic-model-like quality, but rather messages with rare words, particularly those not present in the pre-computed embeddings, and other outliers get singled out while the rest of the messages reside in very large, heterogeneous clusters. The **BLEND$_{WMD}$** measure improves the performance of **WMD** particularly when the number of clusters is small, while behaving similarly as **WMD** when the number of clusters grows. When the number of clusters is small, V-measure somewhat favors **LSA** over **BLEND$_{WMD}$** as can be seen in Figure 6.5. By investigating homogeneity and completeness individually, that is the components of V-measure, we find that homogeneity particularly favors **LSA** while **BLEND$_{WMD}$** wins in completeness. This speaks well for **LSA** since homogeneity is for our purposes more desirable. However, **BLEND$_{WMD}$** overtakes it in homogeneity as well at about 750 clusters. The homogeneity and completeness scores can be seen in detail in Figure 6.6.

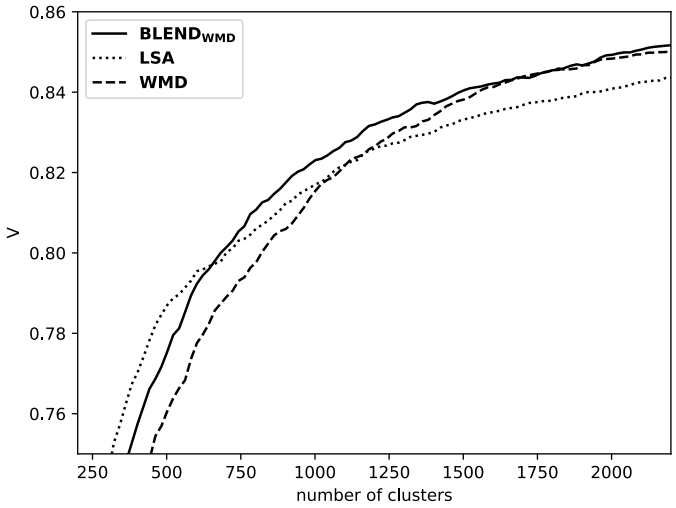

Figure 6.5: The performance of **BLEND$_{\mathbf{WMD}}$**, **LSA**, and **WMD** by V-measure. Homogeneity and completeness are weighted evenly. Note the disagreement with ARI.

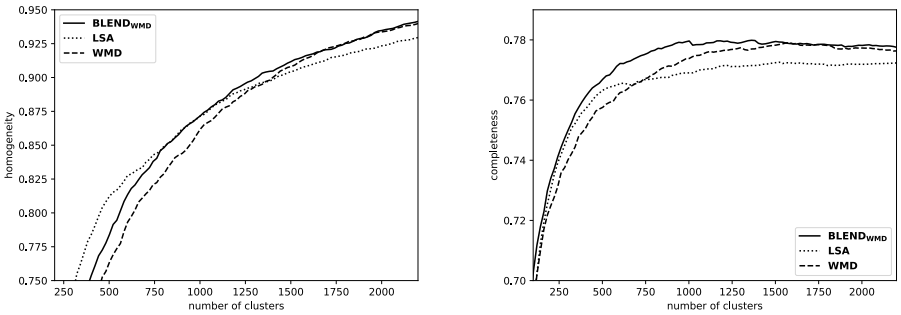

Figure 6.6: The homogeneity (left) and completeness (right) of the clusterings produced by **BLEND$_{\mathbf{WMD}}$**, **LSA**, and **WMD**.

7 Discovering topics

Our primary goal is to extract from a dataset of Slack messages a number of topics, that correctly and informatively represent the contents of the dataset. Conventional topic models, such as LDA, are typically able to produce such topics. However, the messages in the dataset are very short on average, which cause a data sparsity issue. In order to alleviate the sparsity problem, we use a data combination scheme based on certain assumptions about the Slack data. In the previous section, we noted that there are implicit structures in instant messages, and explored the problem of detecting these structures which we call discussions. In this section, the discussions are leveraged in topic modeling as a data combination scheme.

The hypothesis is that LDA will produce a better topic model from the smaller but less sparse dataset of clusters produced by the discussion detection algorithm, than from the raw dataset of individual messages. LDA assumes that the documents are independent of each other, which is not true of instant messages. Instead, instant messages form threads, or discussions, which may be both explicitly and implicitly presents in the data. Since the messages in the same discussion are heavily dependent on each other, and messages in different discussions are not, it is justifiable to segment the text into discussions rather than individual messages. Hypothetically the better segmentation will lead to a better topic model. However, the detection of implicit discussions is not perfect, and will therefore add some noise to the dataset. In the worst case, this may mean that any benefits of the better segmentation are countered by the added noisiness.

7.1 Approach

The models considered in this section are all LDA models, which differ only in the segmentation of the text data. The same preprocessing and tokenization are used. In the preprocessing stage, the data is cleaned by removing emojis and the various formatting tokens used by Slack. In the tokenization step, the messages are divided into unigram tokens, with phrase detection. Very rare and very common terms are filtered out, according to the same criteria which were specified in Section 6.2.1. The input of LDA model training algorithm is a term-document matrix; the term weights are counts.

7.1.1 Baseline model

In order to evaluate the effect of data combination on the quality of model, the models that use data combination are compared to a rudimentary baseline model. The baseline LDA model is trained on the individual messages; that is, each row in the document-term matrix represents a single message. It does not leverage the metadata in any way, nor is the data expanded with any additional information. The baseline model is denoted **BASELINE**.

7.1.2 Discussion-based models

In the *discussion-based models*, The clustering models presented in Section 6 are leveraged as a data combination scheme. LDA is applied to clusters of messages instead of individual messages. First, the data is clustered and word counts for each cluster are counted to form a cluster-term matrix. This is equivalent to concatenating the documents in the clusters to form a new document, and computing a document-term matrix for it. Finally, an LDA model is trained on the cluster-term matrix.

Based on the findings presented in the previous section, the best-performing clustering models, $\mathbf{BLEND_{WMD}}$ and **WMD**, were selected to be used in the topic modeling task. The discussion-based models are denoted in this section by their respective clustering models.

In addition to the number of topics, the discussion-based models also require us to choose the number of clusters. Based on the performance in the discussion detection task, as well as brief experimentation in the topic modeling task, the number of clusters is set to 1000 for the dataset in question. However, the model does not seem to be particularly sensitive to the number of clusters.

7.2 Results

Our ultimate goal is to gain insight on a large dataset of text documents, which is a typical motivation for topic modeling. As discussed in Section 5.2, the ability of a topic model to convey information about the data is often characterized in terms of interpretability and relevance. Human evaluation, as described in Section 5.2.2, would be ideal in such a case, but was ruled out as too laborious. Interpretability can be measured in a fully automated fashion with a high correlation with human judgments using a coherence measure, but relevance can not. This leaves the evaluation presented in this section somewhat incomplete.

Interpretability alone is not a sufficient indicator of quality in this case, since the discussion detection algorithm incorporates external semantic information through the word embeddings. The information in the word vectors is reflected in the resulting topic model, giving cause for concern that the model trades relevance for interpretability. Messages with similar words, as determined by the embeddings, are more likely clustered together, which then causes co-occurrences that are not present in the original data. This is generally desirable, but may be harmful when a word is used in a context specific way which differs from its typical meaning. The word embeddings are learned from a very large and diverse corpus, whereas the datasets we target are from a very specific context (the internal discussions of an IT firm). In some cases this disparity can cause unrelated messages to be mistakenly clustered together, particularly due to synonymy. For instance, the word

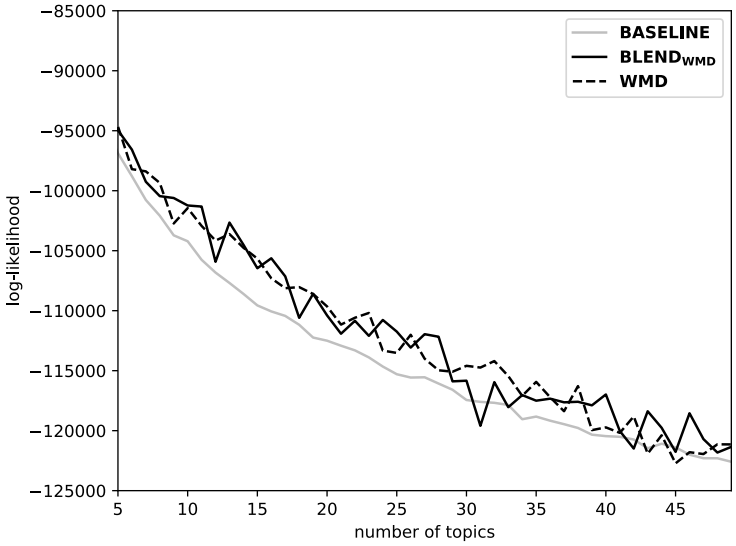

Figure 7.1: The approximate log-likelihood of the models.

mouse quite consistently refers to the device rather than the animal in the kind of data we are dealing with, but a synonymy problem could emerge due to the generic word embeddings. While the data expansion is expected to alleviate the sparsity problem, it may also introduce bias. The bias would not hurt interpretability as the topics taken from the context of the data would be coherent, but as to the actual content of the data they might be irrelevant or even misleading.

We have a clear need to evaluate relevance, but as discussed earlier, fully automated solutions with a high correlation with human judgments do not exist. This leaves us with holdout log-likelihood as the best indicator of overall model quality. The major shortcoming of holdout log-likelihood is that it does not reflect interpretability, but we can measure interpretability separately with a coherence measure.

The approximate log-likelihood of the holdout for the discussion-based models and the baseline model are shown in Figure 7.1. The discussion-based models score somewhat better than the baseline, although they seem to be more sensitive to the number of topics, the **BLEND$_{\mathbf{WMD}}$** model in particular. Note again, that a model's ability to predict unseen data does not necessarily mean the model produces interpretable or meaningful topics.

In Figure 7.2 we see the coherence scores of the **BLEND$_{\mathbf{WMD}}$** and **BASELINE** models, plotted by the number of topics. The coherence measure used is the C_V measure, as presented in Section 5.2.3, which has been found to have the strongest correlation with human judgments. The probabilities are estimated from the English Wikipedia, using the same

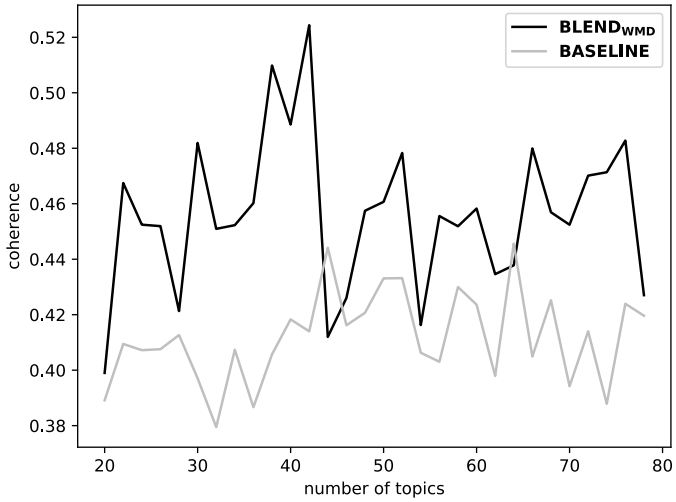

Figure 7.2: The coherence scores of the models, plotted by the number of topics

preprocessing and tokenization algorithms as in the topic model training. The **BLEND$_{WMD}$** model performs clearly better, particularly when the number of topics is lower than 40. However the **BLEND$_{WMD}$** model does not seem to be very robust to the number of topics, as evidenced by the sudden drop in performance at about 45 topics. The **BASELINE** model fluctuates less wildly; this can be seen with approximate log-likelihood as well.

In Figures 7.3 and 7.4, the top topics produced by the **BASELINE** and **BLEND$_{WMD}$** models are presented. 10 largest topics by marginal probability are shown for each model, and for each topic, 10 most probable words are shown along with their relative probabilities in the topic-word distribution. The individual topics for each model are denoted BASELINE#1 to BASELINE#10 and BLEND#1 to BLEND#10 respectively. One can immediately see that neither the topics produced by the **BASELINE** nor the **BLEND$_{WMD}$** model are particularly easily interpretable. Some of the topics produced by **BLEND$_{WMD}$** are interpretable, however. The topic BLEND#7 is clearly related to web development, and the topic BLEND#8 contains terminology related to frontend development along with unrelated terms. BLEND#4 is recognisably about business intelligence and analytics, and includes the terms *hohto* and *näe_gofore* which refer to relevant internal projects. BLEND#2 seems to refer to a particular customer project. BLEND#9 seems to be a composite of two distinguishable topics – *shirt*, *shirts*, and *seppo* (a company mascot) refer to marketing materials, while

salts, *compromised*, and perhaps *memory*, point towards computer security. The other topics are less recognisable.

In summary, the discussion-based model improved the coherence of the topic model, as well as its predictive power when measured by holdout likelihood. We may conclude that the discussion detection based approach produced a more interpretable model, as indicated by increased C_V coherence. We may also conclude that the discussion based approach has benefits beyond increased interpretability, which suggests that the increased coherence does not come entirely from a tradeoff between interpretability and relevance. However, the overall quality of the discussion-based topic model is not very good in objective terms, as demonstrated by the topics in Figure 7.4. Due to the low interpretability of the topics they are not very useful in summarizing or gaining insight on the data.

7.3 Summarizing the dataset using clustering

Ideally we would have wanted a moderate number of easily interpretable topics that could be represented as sets of words. However, the quality of even the best topic model proved to be too low for such summaries to be understandable. The clustering alone proved to be more useful for summarizing the data. This section briefly discusses using the clustering presented in Section 6 to produce a summarizing visualization of the Slack dataset.

Although clustering algorithms do not produce topics per se, it is possible to use clustering for similar purposes as topic modeling. They key difference is that cluster assignments are hard, and so a cluster always represents only a specific part of the dataset, while topics may represent more general aspects of the data. Hierarchical clustering has the benefit of producing a cluster tree instead of a flat clustering. The whole tree can be leveraged in visualization.

Figure 7.5 shows a screenshot of a visualization tool that was implemented on the basis of the $\mathbf{BLEND_{WMD}}$ clustering model. It presents the hierarchical clustering using a visualization known as a circle pack or a circular treemap. The original binary tree produced by the clustering algorithm has been flattened so that each level has an appropriate number of clusters for the visualization. Each circle in the circle pack represents a cluster, and the size of the circle is relative to the size of the cluster. The user may zoom into a cluster to see its sub-clusters, all the way down to the individual messages.

The most significant problem with the clustering approach is that there is no natural way of presenting the cluster contents. One might use statistics like term frequencies or TF-IDFs, but these do not accurately reflect why the documents were clustered as they were, and may therefore be misleading. Sharing terms will make documents more likely to be clustered together, but so will similar but distinct words. A term that does not appear frequently in

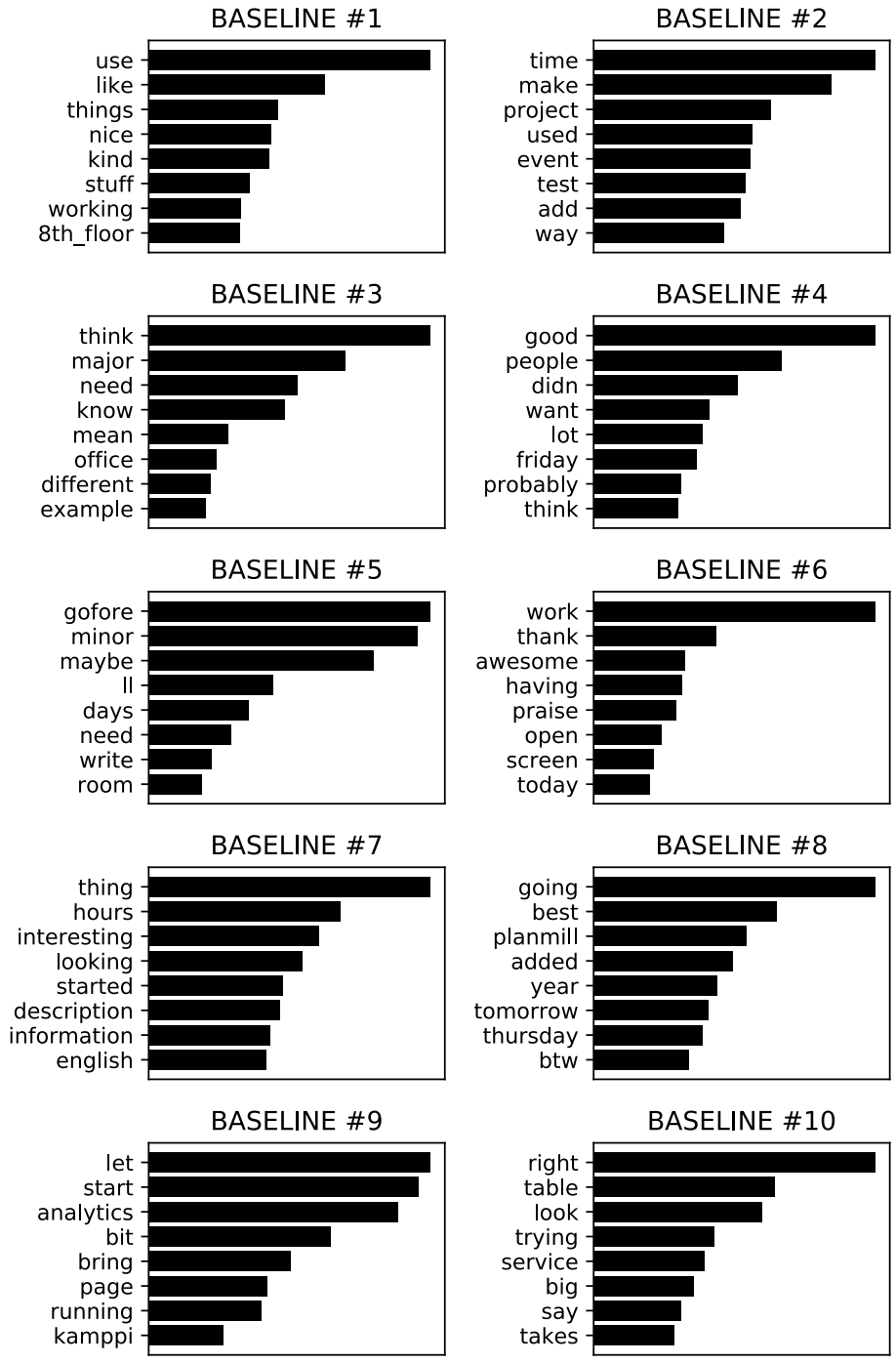

Figure 7.3: The top topics produced by the BASELINE model, represented by their most probable words.

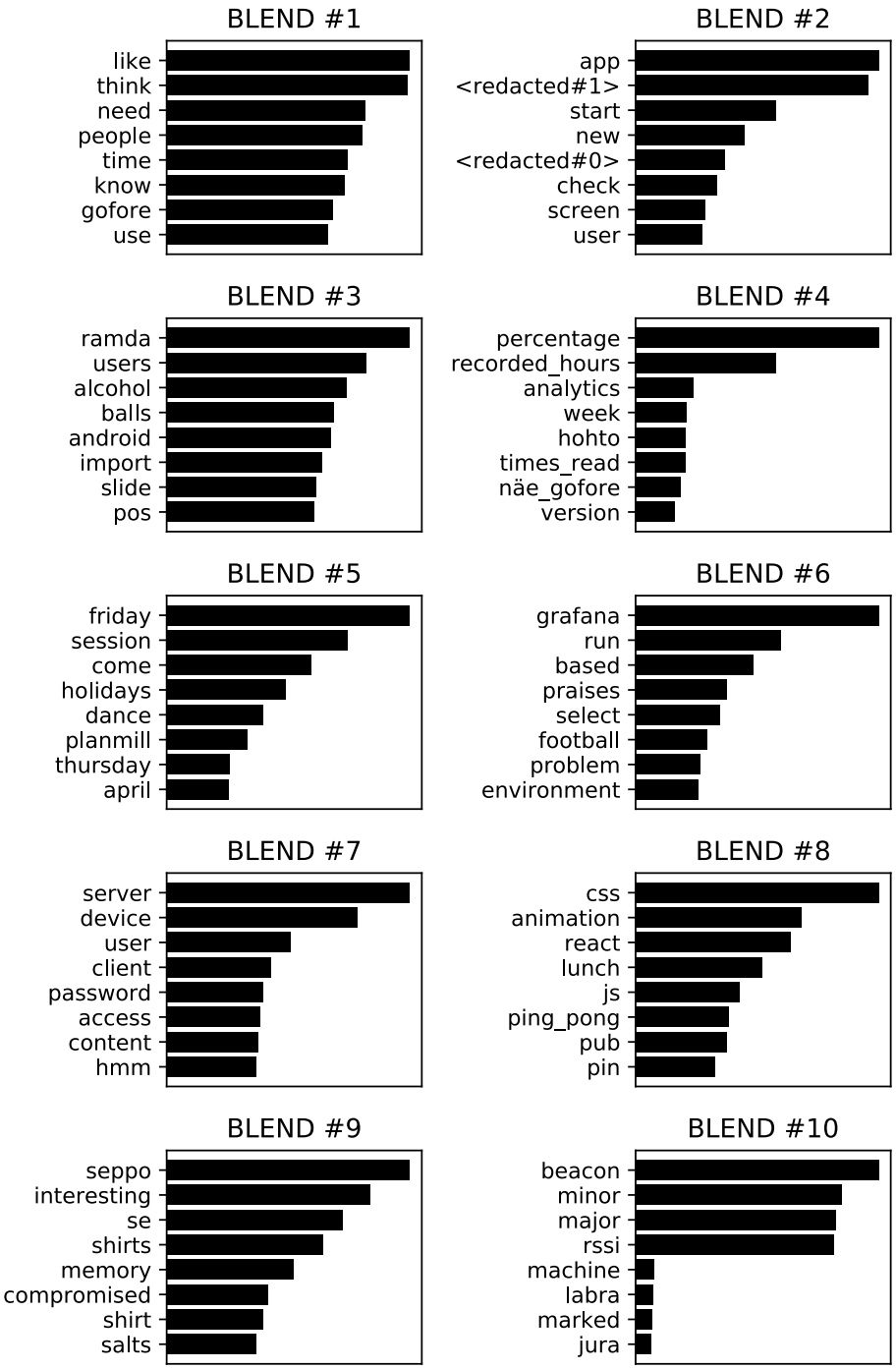

Figure 7.4: The top topics produced by the BLEND model.

Figure 7.5: A screenshot of the visualization tool that was implemented using the clustering algorithm presented in Section 6. Pictured is a single cluster with subclusters and their word cloud summaries. (Note that some words have been redacted, hence the black blocks.)

a cluster as a whole may still have a strong effect on a particular document being assigned to that cluster. Simple term statistics may therefore make a cluster look far more specific than it actually is.

In the visualization shown in Figure 7.5, cluster contents are summarized using word clouds, where the size and color of a word is relative to the word's relevance to the cluster. The relevance is based on the aggregate weights of the terms in the cluster. To alleviate the aforementioned issues with summarizing a cluster in such a way, the term weights are scaled according to the term's similarity to a mean vector using the word embeddings. This is an ad hoc solution however, and the choice over using simply the aggregate term weights, word frequencies, or any other method is simply a matter of taste. The clustering algorithm is based purely on the document distances, and lacks any knowledge of the individual terms. Therefore estimating the effects of individual terms on the clustering is a complicated affair.

The clustering and topic modeling approaches are rather different, and a proper comparison between the approaches would have to be done through user studies, which is beyond the scope of this thesis. The two approaches should be seen as complementary, particularly as the clustering is useful in improving the topic model as well.

8 Conclusions

Data from the Slack instant messaging platform is a potential source of valuable insight, but the data is difficult to analyze using typical NLP techniques. This is due to data sparsity issues caused by the shortness of typical Slack messages. We noted however, that there are both explicit and implicit structures in the Slack data which may be leveraged to alleviate the issues caused by data sparsity.

The purpose of this thesis was to discover topics in a dataset of Slack messages. Conventional topic modeling methods run into trouble with sparse data however, as evidenced by the poor performance of the baseline model presented in Section 7. The proposed solution involves an auxiliary task of detecting discussions, or implicit threads, in the message stream. A solution based on hierarchical clustering and word embeddings was introduced in Section 6.

An LDA topic model was then trained on the clusters. The discussion-based topic model showed improvement over the baseline model in coherence and approximate holdout log-likelihood. However, the overall quality of the topics remained low. A purely clustering-based approach for summarizing was presented as a complementary solution.

References

[1] Adams, P. H. and Martell, C. H.: *Topic Detection and Extraction in Chat.* In *2008 IEEE International Conference on Semantic Computing*, pages 581–588, August 2008.

[2] Aggarwal, Charu C. and Zhai, ChengXiang: *A Survey of Text Clustering Algorithms.* In *Mining Text Data*, pages 77–128. Springer, Boston, MA, 2012.

[3] Allahyari, Mehdi, Pouriyeh, Seyedamin, Assefi, Mehdi, Safaei, Saied, Trippe, Elizabeth D., Gutierrez, Juan B., and Kochut, Krys: *A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques.* arXiv:1707.02919 [cs], July 2017. `http://arxiv.org/abs/1707.02919`.

[4] Asuncion, Arthur, Welling, Max, Smyth, Padhraic, and Teh, Yee Whye: *On Smoothing and Inference for Topic Models.* In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 27–34, Arlington, Virginia, United States, 2009. AUAI Press. event-place: Montreal, Quebec, Canada.

[5] Becker, Hila: *Identification and Characterization of Events in Social Media.* PhD Thesis, Columbia University, New York, NY, USA, 2011.

[6] Blei, David M.: *Probabilistic Topic Models.* Communications of the ACM, 55(4):77–84, April 2012, ISSN 0001-0782.

[7] Blei, David M., Ng, Andrew Y., and Jordan, Michael I.: *Latent Dirichlet Allocation.* Journal of Machine Learning Research, 3:993–1022, March 2003, ISSN 1532-4435.

[8] Bojanowski, Piotr, Grave, Edouard, Joulin, Armand, and Mikolov, Tomas: *Enriching Word Vectors with Subword Information.* Transactions of the Association for Computational Linguistics, 5:135–146, 2017.

[9] Campello, Ricardo J. G. B., Moulavi, Davoud, and Sander, Joerg: *Density-Based Clustering Based on Hierarchical Density Estimates.* In Pei, Jian, Tseng, Vincent S., Cao, Longbing, Motoda, Hiroshi, and Xu, Guandong (editors): *Advances in Knowledge Discovery and Data Mining*, pages 160–172, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[10] Chang, Jonathan, Gerrish, Sean, Wang, Chong, Boyd-graber, Jordan L., and Blei, David M.: *Reading Tea Leaves: How Humans Interpret Topic Models.* In *Advances in Neural Information Processing Systems 22*, pages 288–296. Curran Associates, Inc., 2009.

[11] De Boom, Cedric, Van Canneyt, Steven, Demeester, Thomas, and Dhoedt, Bart: *Representation learning for very short texts using weighted word embedding aggregation.* Pattern Recognition Letters, 80:150–156, September 2016, ISSN 01678655.

[12] Deerwester, Scott, Dumais, Susan T., Furnas, George W., Landauer, Thomas K., and Harshman, Richard: *Indexing by latent semantic analysis.* Journal of the American Society for Information Science, 41(6):391–407, September 1990, ISSN 0002-8231, 1097-4571.

[13] Ester, Martin, Kriegel, Hans Peter, Sander, Jörg, and Xu, Xiaowei: *A density-based algorithm for discovering clusters in large spatial databases with noise.* In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.

[14] Flisar, Jernej and Podgorelec, Vili: *Document Enrichment Using DBPedia Ontology for Short Text Classification.* In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, WIMS '18, pages 8:1–8:9, New York, NY, USA, 2018. ACM.

[15] Gardner, A., Duncan, C. A., Kanno, J., and Selmic, R. R.: *On the Definiteness of Earth Mover's Distance and Its Relation to Set Intersection.* IEEE Transactions on Cybernetics, 48(11):3184–3196, November 2018, ISSN 2168-2267.

[16] Grave, Edouard, Bojanowski, Piotr, Gupta, Prakhar, Joulin, Armand, and Mikolov, Tomas: *Learning Word Vectors for 157 Languages.* In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[17] Han Lau, Jey, Newman, David, and Baldwin, Timothy: *Machine Reading Tea Leaves: Automatically Evaluating Topic Coherence and Topic Model Quality.* 14th Conference of the European Chapter of the Association for Computational Linguistics 2014, EACL 2014, pages 530–539, 2014.

[18] Hubert, Lawrence and Arabie, Phipps: *Comparing partitions.* Journal of Classification, 2(1):193–218, December 1985, ISSN 1432-1343.

[19] Kusner, Matt J., Sun, Yu, Kolkin, Nicholas I., and Weinberger, Kilian Q.: *From Word Embeddings to Document Distances.* In *Proceedings of the 32nd International Conference on Machine Learning - Volume 37*, ICML'15, pages 957–966, Lille, France, 2015.

[20] Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey: *Efficient Estimation of Word Representations in Vector Space.* arXiv:1301.3781 [cs], January 2013. `http://arxiv.org/abs/1301.3781`.

[21] Mikolov, Tomas, Grave, Edouard, Bojanowski, Piotr, Puhrsch, Christian, and Joulin, Armand: *Advances in Pre-Training Distributed Word Representations.* Proceedings of the Eleventh International Conference on Language Resources and Evaluation, 2018.

[22] Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg, and Dean, Jeffrey: *Distributed Representations of Words and Phrases and their Compositionality.* arXiv:1310.4546 [cs, stat], October 2013. `http://arxiv.org/abs/1310.4546`.

[23] Pangos, A., Iosif, E., Potamianos, A., and Fosler-Lussier, E.: *Combining statistical similarity measures for automatic induction of semantic classes.* In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pages 278–283, November 2005.

[24] Pfitzner, Darius, Leibbrandt, Richard, and Powers, David: *Characterization and evaluation of similarity measures for pairs of clusterings.* Knowledge and Information Systems, 19(3):361, July 2008, ISSN 0219-3116.

[25] Rand, W.M.: *Objective criteria for the evaluation of clustering methods.* Journal of the American Statistical Association, 66(336):846–850, 1971.

[26] Romano, Simone, Vinh, Nguyen Xuan, Bailey, James, and Verspoor, Karin: *Adjusting for Chance Clustering Comparison Measures.* Journal of Machine Learning Research, 17(1):4635–4666, January 2016, ISSN 1532-4435.

[27] Rosenberg, Andrew and Hirschberg, Julia: *V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure.* In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning(EMNLP-CoNLL)*, pages 410–420, 2007.

[28] Rubner, Y., Tomasi, C., and Guibas, L. J.: *A metric for distributions with applications to image databases.* In *Sixth International Conference on Computer Vision*, pages 59–66, January 1998.

[29] Röder, Michael, Both, Andreas, and Hinneburg, Alexander: *Exploring the Space of Topic Coherence Measures.* In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, pages 399–408, New York, NY, USA, 2015. ACM. event-place: Shanghai, China.

[30] Sahami, Mehran and Heilman, Timothy D.: *A Web-based Kernel Function for Measuring the Similarity of Short Text Snippets.* In *Proceedings of the 15th International Conference on World Wide Web*, WWW '06, pages 377–386, New York, NY, USA, 2006. ACM, ISBN 1-59593-323-9.

[31] Sahlgren, Magnus: *The distributional hypothesis.* Italian Journal of Linguistics, 20, 2008.

[32] Salton, G., Wong, A., and Yang, C. S.: *A Vector Space Model for Automatic Indexing.* Communications of the ACM, 18(11):613–620, November 1975, ISSN 0001-0782.

[33] Shen, Dou, Yang, Qiang, Sun, Jian Tao, and Chen, Zheng: *Thread Detection in Dynamic Text Message Streams.* In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 35–42, New York, NY, USA, 2006. ACM.

[34] Spärck Jones, Karen: *A Statistical Interpretation of Term Specificity and its Application in Retrieval.* Journal of Documentation, 28:11–21, December 1972.

[35] Vinh, Nguyen Xuan, Epps, Julien, and Bailey, James: *Information Theoretic Measures for Clusterings Comparison: Is a Correction for Chance Necessary?* In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 1073–1080, New York, NY, USA, 2009. ACM.

[36] Weng, Jianshu, Lim, Ee Peng, Jiang, Jing, and He, Qi: *TwitterRank: Finding Topic-sensitive Influential Twitterers.* In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 261–270, New York, NY, USA, 2010. ACM.